PROPOSE.AI

Projektdokumentation

Vorgelegt von

Robert Albrecht
Felix Jedebrock
Lea Lohmann
Wiebke Meyer
Jannis Oeltjen
Jannik Otten
Michael Riedel
Jonas Tiemerding
Jan-Hendrik Witte
Stefan Zurborg

2018/2019

In Zusammenarbeit mit

Carl von Ossietzky Universität Oldenburg

Abteilung Wirtschaftsinformatik/ Very Large Business Applications

BRILLE24
Research

Online Optiker: Brille24

Kontakt:

Themensteller:	Prof. Dr.-Ing. Jorge Marx Gómez
Betreuer VLBA:	M.Sc. Marius Wybrands
Betreuer VLBA:	M.Sc. René Kessler
Betreuer Brille24:	M.Sc. Manuel Zapp

Gruppenmitglieder:

Robert Albrecht	robert.albrecht@uni-oldenburg.de
Felix Jedebrock	felix.jedebrock@uni-oldenburg.de
Lea Lohmann	lea.lohmann@uni-oldenburg.de
Wiebke Meyer	wiebke.meyer@uni-oldenburg.de
Jannis Oeltjen	jannis.oeltjen@uni-oldenburg.de
Jannik Otten	jannik.otten@uni-oldenburg.de
Michael Riedel	michael.riedel@uni-oldenburg.de
Jonas Tiemerding	jonas.tiemerding@uni-oldenburg.de
Jan-Hendrik Witte	jan-hendrik.witte@uni-oldenburg.de
Stefan Zurborg	stefan.zurborg@uni-oldenburg.de

CvO Universität Oldenburg
Ammerländer Heerstraße 114-118
26129 Oldenburg

Brille24 GmbH
Alter Stadthafen 10
26122 Oldenburg

Danksagung

An dieser Stelle möchten wir die Gelegenheit nutzen, um Danke zu sagen. Unser Dank gilt allen voran der Abteilung Very Large Business Applications (VLBA) der Uni Oldenburg und unserem Projektpartner Brille24. Hierbei bedanken wir uns besonders bei Prof. Dr.-Ing. Jorge Marx Gómez und Christophe Hocquet, ohne die dieses Projekt nicht möglich gewesen wäre.

Außerdem möchten wir uns beim Team von Brille24 für die tatkräftige Unterstützung bedanken, namentlich genannt seien an dieser Stelle Patrick Jürgens, Jens Rütter, Christian Arndt, Conny Wiecker und Meike Rüppel.

Unser besonderer Dank gilt Marius Wybrands, René Kessler und Manuel Zapp für die hervorragende Betreuung und die Unterstützung auch in schwierigen Phasen des Projektes.

Inhaltsverzeichnis

Abkürzungsverzeichnis

API Application-Programming-Interface

CBOR Concise Binary Object Representation

CTPN Connectionist Text Proposal Network

JSON JavaScript Object Notation

LSTM Long short-term memory

MIME Multipurpose Internet Mail Extensions

MSER Maximally Stable Extremal Regions

POSIX Portable Operating System Interface

REST Representational State Transfer

UUID Universally Unique Identifier

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die vorliegende Arbeit wurde im Rahmen der Projektgruppe Propose.AI angefertigt und dokumentiert die Arbeitsergebnisse sowie den Projektablauf. Das Projekt wurde an der Carl von Ossietzky Universität Oldenburg in der Abteilung Very Large Business Applications (VLBA) unter der Leitung von Prof. Dr.-Ing. habil. Jorge Marx Gómez und in Kooperation mit dem Oldenburger Onlineoptiker Brille24 im Zeitraum April 2018 bis April 2019 durchgeführt. Abschnitt 1.1 enthält die Motivation und Zielsetzung des Projektes, 1.2 die Vorstellung des Praxispartners Brille24 GmbH.

1.1 Motivation und Zielsetzung

Die Regeln des Handels wurden in den letzten Jahren kontinuierlich neu definiert. Neben dem stationären Handel ist es vor allem der Onlinehandel, welcher in den vergangenen Zeitabschnitten starken Veränderungen ausgesetzt worden ist. Kaum eine Sparte ist einem ähnlichen Konkurrenzdruck ausgesetzt wie der Onlinehandel. Dies spiegelt sich nicht nur in den entsprechenden Zahlen wider. Allein bei Betrachtung der Entwicklung des Umsatzes im E-Commerce der letzten zehn Jahre fällt auf, dass in diesem Segment ein enormes Wachstumspotential besteht und es derzeit keinerlei Anzeichen für eine Änderung dieses Trends in den kommenden Jahren gibt. Innerhalb von nur zehn Jahren konnte der Umsatz von ca. 15,6 Milliarden Euro in 2009 auf unglaubliche 58,5 Milliarden Euro in 2019 wachsen, Tendenz steigend [Sta19]. Nicht nur, dass dies eine nahezu Vervierfachung des Umsatzes darstellt, sowohl der stationäre Handel als auch der Onlinehandel sind aufgrund des kontinuierlichen Wachstums einem starken Konkurrenzdruck ausgesetzt. Immer mehr Unternehmen bieten ihre Produkte und Dienstleistungen über das Internet an oder versuchen, diese auf den E-Commerce zu übertragen. Durch die neuen Möglichkeiten des E-Commerce entstehen jedoch auch grundlegend neue, innovative Geschäftsmodelle, welche teilweise disruptive Auswirkungen auf einzelne Marktsegmente haben können. Bereits 2014 zeigte eine Statistik, dass ca. 23% aller Unternehmen ihre Waren sowie Dienstleistungen ebenfalls über das Internet anbieten, was nochmals die Relevanz und Wichtigkeit dieses Marktes verdeutlicht [Man17].
Es existieren viele unterschiedliche Faktoren die Kunden dazu bewegen, gewünschte Produkte online zu kaufen. Neben der erhöhten Bequemlichkeit und Einfachheit, der größeren Auswahl und des oftmals günstigeren Preises gibt es noch weitere Einflüsse, die diesen Trend der letzten Jahre weiter bestärken. Das Fashion- und Modesegment des E-Commerce ist dabei jedoch sowohl ein interessanter als auch komplizierter Anwendungsfall. Vor al-

lem im zuvor genannten Segment existieren teils erhebliche Diskrepanzen zwischen dem Einkaufserlebnis des Online- und stationären Handels.
Zwar liegen im Bezug auf das Segment Fashion noch immer die zentralen Vorteile des E-Commerce gegenüber dem stationären Handel vor, wie die erhöhte Bequemlichkeit, die größere Auswahl und der oftmals günstigere Preis, allerdings herrschen starke Gegenfaktoren, die einen Kunden vor einem potenziellen Wechsel zum E-Commerce hindern könnten. Diese Faktoren spiegeln sich sowohl in der Möglichkeit der Inanspruchnahme einer persönlichen Beratung durch den Mitarbeiter vor Ort wider als auch durch die Möglichkeit einer unmittelbaren Anprobe des jeweilig betrachteten Objekts. Durch die lokale Beratung im stationären Handel ist dem Kunden eine Auswahl personalisierter Produktvorschläge garantiert, die, je nach Kompetenz der jeweiligen Fachkraft, sowohl unter Berücksichtigung modischer Aspekte als auch bezogen auf die entsprechenden Maße des Kunden speziell für ihn ausgewählt wurden. Zwar bieten Onlineshops ebenfalls die Möglichkeit zu „personalisierten“Produktvorschlägen auf Basis von bspw. Recommendation-Engines, allerdings erreichen diese nicht die Qualität einer persönlichen Beratung.
Der Praxispartner des Projekts befindet sich in einer ähnlichen Situation. Die zuvor erwähnten Vorteile des E-Commerce greifen auch beim angewendeten Geschäftsmodell von Brille24, allerdings ist auch hier die Möglichkeit der persönlichen Beratung des stationären Handels ein wesentlicher Vorteil gegenüber des Onlinehandels. Um den Onlinekauf von Brillen für den Kunden attraktiver zu gestalten, soll eine ähnliche persönliche Beratung im Onlinehandel ermöglicht werden, welche ähnliche Qualitätsmerkmale erfüllt. Aus dieser Intention wurde ein erstes Grobkonzept in Form eines Anwendungsfalls formuliert, der Gestaltung und Entwicklung einer AI-Service Plattform zur personalisierten Produktempfehlung auf Basis heterogener Daten. Ziel ist es, einen Service auf Basis künstlicher Intelligenz zu entwickeln, welcher Kunden auf Grundlage eines Fotos ihres Gesichts personalisierte Brillenvorschläge generiert, um die persönliche Beratung des stationären Handels im E-Commerce zu simulieren und nachzubilden. Im Verlauf der Projektgruppe ist aus dem anfänglich formulierten Grobkonzept ein komplexes Konstrukt unterschiedlichster Services und Technologien entstanden. Durch die stetige Überarbeitung, Veränderung, Weiterentwicklung und Ergänzung des Konzepts durch neue Ideen und Gedanken konnte eine Ansammlung unterschiedlichster Services erschlossen werden, welche im Rahmen dieser Dokumentation vorgestellt werden sollen. Zwar war die ursprünglich formulierte Problemstellung und die daraus resultierende Intention über den gesamten Projektverlauf von Relevanz, allerdings wurden auf Grundlage neu identifizierter Problemstellungen im Verlauf des Projektes weitere Teilprojekte hinzugefügt, auf welche im späteren Verlauf dieses Dokuments eingegangen wird.

1.2 Unternehmensportrait: Brille24

Die Brille24 GmbH ist ein Online-Optiker mit Sitz in Oldenburg. Das Unternehmen wurde im Jahr 2007 u. a. durch Matthias Hunecke, Oliver Klapproth und Tim Gebken gegründet und ging im Jahr 2008 online. Mit über zwei Millionen verkauften Brillen ist die Brille24 GmbH einer der größten Online-Optiker im deutschsprachigen Raum und ist neben Deutschland auch in Belgien, Frankreich, den Niederlanden, Portugal sowie Spanien aktiv und hat über eine Million registrierte Kunden in 117 Ländern. Angefangen mit Brillen von Eigenmarken hat Brille24 seit März 2016 auch Modelle namhafter Hersteller im Sortiment. Insgesamt bietet Brille24 rund 4700 Brillenmodelle an, wovon 3500 von namhaften Herstellern direkt bezogen werden. Das Portfolio umfasst Einstärkenbrillen, Gleitsichtbrillen, Sonnenbrillen, Sportbrillen und auch Skibrillen. Des Weiteren verkauft Brille24 auch Kontaktlinsen sowie Brillen mit selbsttönenden oder polarisierenden Gläsern [Bri18].

Seit 2017 ist Brille24 im Bereich der künstlichen Intelligenz aktiv und hat, um die Forschung voranzutreiben, unter dem Namen „Brille24 Research“ eine KI-Abteilung eingerichtet. Das Unternehmen arbeitet in diesem Bereich mit Kooperationspartnern wie der Universität Oldenburg und dem Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI) zusammen. Ziel dieser Zusammenarbeit ist der Einsatz von künstlicher Intelligenz im Vertrieb und Service sowie besonders im Marketing [Mey18].

2 Projektmanagement

In diesem Abschnitt wird das Projektmanagement der Projektgruppe näher erläutert. Zunächst werden mit Scrum und Kanban die Vorgehensmodelle der agilen Softwareentwicklung beschrieben, die Anwendung fanden. Des Weiteren wird auf die einzelnen Projektphasen eingegangen als auch auf die Rollenverteilung sowie die Rollenbeschreibung.

2.1 Agile Softwareentwicklung

Bei der agilen Softwareentwicklung geht es darum, den Entwicklungsprozess möglichst transparent und flexibel zu gestalten, um einen schnelleren Einsatz der zu entwickelten Systeme zu gewährleisten. Dabei sollen die Teilprozesse möglichst einfach und somit agil gehalten werden. Durch einen stetigen Kundenkontakt, welcher zu verschiedenen Zeitpunkten des Entwicklungsprozesses Einsicht in die bereits erstellen Produktsegmente erhält, und die dadurch resultierende Möglichkeit auf Änderungswünsche einzugehen, wird das Risiko minimiert, ein falsches Produkt für den Kunden zu entwickeln [Sie18]. Dabei gibt es verschiedene Prozesse zur Umsetzung der agilen Softwareentwicklung, wobei im Weiteren nur auf Scrum und Kanban eingegangen wird, da diese von der Projektgruppe als am sinnvollsten für das Projekt eingeschätzt wurden.

2.1.1 Scrum

Der wohl geläufigste Prozess der agilen Softwareentwicklung ist Scrum. Dabei soll durch den Prozess die Produktivität und die Zufriedenheit der Entwickler sowie die Codequalität und Transparenz gesteigert werden. Hierzu werden die Entwickler in kleinere Teams aufgeteilt, die aus einem Entwicklerteam, einem Scrum Master und einem Product Owner bestehen. Das Team arbeitet dabei in Sprints, die eine vorgegebene Dauer haben und in denen ein fertiges Produktinkrement entstehen soll, das dem Kunden vorgelegt werden kann. Ein Sprint besteht dabei aus:

- Planungsmeeting – in dem festgelegt wird, was das Team in dem Sprint schaffen möchte
- Sprint selber – in dem die Entwickler die gegebenen Aufgaben erledigen
- Reviewmeeting – in dem das Inkrement vorgestellt wird
- Retrospektive – in der das Team diskutiert, welche Dinge gut oder schlecht in dem Sprint gelaufen sind

Zudem wird jeden Tag ein Daily Scrum ausgeführt, in dem jeder aus dem Entwicklerteam kurz erläutert, was er/sie den letzten Tag geschafft hat, wo Probleme waren und wo weiterhin Hindernisse bestehen [Glo10].

Der Product Owner ist dafür zuständig, ein Product Backlog zu erstellen, in dem die verschiedenen Aufgaben, die für die Erstellung des Produktes erforderlich sind, in Form von User Stories aufgeführt werden. Das Team kann sich dementsprechend beim Planungsmeeting die entsprechenden User Stories aus dem Product Backlog nehmen, die für den Sprint als relevant betrachtet werden. Das Product Backlog kann dabei zu jeder Zeit je nach Kundenwunsch oder wenn Bugs auftauchen erweitert werden. Der Scrum Muster hat dabei die Aufgabe, die einzelnen Schritte des Sprints anzuleiten und zu betreuen. Zudem muss er dafür sorgen, dass Team von den anderen Parteien abzuschirmen, damit dieses sich alleine auf die Entwicklung konzentrieren kann [Ham09].

2.1.2 Kanban

Bei Kanban handelt es sich ebenfalls um eine Vorgehensweise der schlanken bzw. agilen Softwareentwicklung. Wie auch bei Scrum wird dabei ein Product Backlog mit allen Aufgaben, die zu erledigen sind, angefertigt. Dabei sollen die Aufgaben nach einem Pull-System aus den einzelnen Arbeitsschritten von dem Entwicklerteam entnommen werden, damit dieses selbst entscheiden kann, wann genügend Kapazitäten für die Bearbeitung vorhanden sind. Um die Übersicht zu behalten, wird ein Kanban-Board verwendet, auf dem das Product Backlog und die einzelnen Phasen abgebildet sind und auf welchem dann die Aufgaben, wie bei Scrum, verschoben werden können. Bei dem ganzen Prozess gilt es generell, dass zunächst der Geschäftswert für den Auftraggeber, dann der gleichmäßige Arbeitsfortschritt und erst dann die Beseitigung von Ballast, der den Arbeitsfortschritt verlangsamt, im Vordergrund steht. So werden Aufgaben, die eine hohe Priorität besitzen, ganz nach oben gesetzt. Das Team soll dementsprechend immer die Aufgaben bearbeiten, die ganz oben aufgelistet werden, da diese den größten Mehrwert für den Auftraggeber erzielen. Dabei sind die Aufgaben, die sich gleichzeitig in der Wertschöpfungskette befinden, in der Anzahl limitiert, was eine Überlastung der Bearbeiter vermeidet und für einen schnellen Abschluss der Aufgaben sorgen soll [Epp11].

2.2 Rollenverteilung

Zu Beginn des Projektes wurden einige Rollen und deren Aufgabenbereiche festgelegt. Diese wurden dann den Projektmitgliedern zugewiesen. Einige Aufgaben waren dabei über

den gesamten Projektverlauf festgesetzt, bei anderen wechselten die Zuständigkeiten. Eine Auflistung der einzelnen Rollen mit verantwortlichen Personen ist in Tabelle 2.2 zu sehen.

Rolle	**Verantwortliche/r**	**Projektphase**
Projektleitung & Projektmanagement	Lea Lohmann Wiebke Meyer	Phase 1
Projektleitung	Lea Lohmann Wiebke Meyer	ab Phase 2
Projektmanagement	Michael Riedel	ab Phase 2
Scrum Master	Felix Jedebrok	Phase 1
Dokumentationsbeauftragter	Jonas Tiemerding Michael Riedel	Phase 1 ab Phase 2
Systemadministrator	Jannis Oeltjen Jannik Otten	alle Phasen
Blog-Betreuung	Stefan Zuborg Jan-Hendrik Witte Jonas Tiemerding	alle Phasen
Strafenbeauftragter	Jan-Hendrik Witte	alle Phasen

Tabelle 1: Rollenverteilung

Projektleitung

Nach der DIN-Norm 69901 ist die Projektleitung eine für die Projektdauer geschaffene Organisationseinheit, welche die Verantwortlichkeit für die Planung, Steuerung und Überwachung des Projektes trägt. Es besteht im Rahmen der jeweiligen Phasen des Projekts die Möglichkeit, die Verantwortlichkeiten an bestehende Bedürfnisse anzupassen. Über den Zeitraum der Projektdauer ist der Projektleiter die zentrale Instanz, welche dem Projektauftraggeber direkten Bericht erstattet [WM08, S.37].

Im Rahmen der Projektgruppe wurde die Rolle des Projektleiters durch Organisationsaufgaben definiert. Dazu gehörten die Planung von Terminen, wie Meetings und Vorträgen, die Kommunikation sowohl mit Betreuern als auch dem Praxispartner, sowie die Vorbereitung und Moderation der Meetings. Ein weiterer Aufgabenbereich, um den sich die Projektleitung gekümmert hat, war die Organisation von Social-Events, wie z.B. einem Fußballturnier. Für die Verantwortlichkeit wurden zwei Projektmitglieder zu Beginn ausgewählt und für den gesamten Zeitraum des Projektes eingesetzt.

Projektmanagement

In der Praxis werden die Leitung eines Projektes und das Projektmanagement häufig

synonym verstanden [Zel17, S.1]. In der Projektgruppe wurden diese beiden Bereiche aufgeteilt. Das Projektmanagement war für die in der Definition aufgeführten Aufgaben der Projektleitung zuständig, die von dieser selbst nicht übernommen wurden. Dabei handelte es sich vor allem um das Erstellen von Projektplänen und die Überwachung von Meilensteinen und Fristen. Diese Rolle wurde nach der ersten Phase von der Projektleitung getrennt und von einem anderem Projektmitglied übernommen.

Scrum Master

In der ersten Phase des Projektes wurde Scrum als Vorgehensmodell eingesetzt. Der Scrum Master war zu Beginn dafür zuständig, die Grundlagen von Scrum und ein Konzept für die Umsetzung in der Projektgruppe vorzustellen. Im weiteren Verlauf hätte der Scrum Master die Meetings geleitet, das Scrum Board gepflegt und abschließende Retrospektiven durchgeführt. Im Laufe der ersten Phase hat sich aufgrund der Aufteilung in einzelne Kleingruppen herausgestellt, dass Scrum für das Projekt nicht sinnvoll war. Folglich wurde diese Rolle aufgelöst.

Dokumentationsbeauftragter

Im Laufe des Projekts gab es zwei Dokumentationsbeauftragte. Für die erste Phase war das Projektmitglied mit dieser Rolle dafür zuständig das Dokument für die Dokumentation aufzusetzen, zu gliedern, zugänglich zu machen und Richtlinien für das gemeinsame Schreiben festzuhalten. In der zweiten Phase wurde die Verantwortlichkeit der Doku an ein anderes Projektmitglied übergeben, womit sich auch neue Aufgaben ergeben hatten. Diese bestanden darin, den Fortschritt der Dokumentation zu überwachen und anzuleiten. Ebenfalls musste von dem Dokumentationsbeauftragten die Einhaltung der Richtlinien überprüft und ggf. angepasst werden.

Systemadministrator

Die Systemadministratoren sind verantwortlich gewesen für die Planung, den Aufbau und die Pflege des Build- und Testsystems. Zusätzlich fiel die Verwaltung des Ticketsystems in den Aufgabenbereich der Systemadministratoren.

Strafenbeauftragter

Zu Beginn des Projektes wurden bestimmte Verhaltensweisen und Regeln festgelegt. Dazu gehörte z. B. das pünktliche Erscheinen zu den Meetings. Wurden diese Regeln nicht eingehalten, sollte je nach Verstoßausmaß eine Strafe erhoben werden. Der Strafbeauftragte hatte die Aufgabe, Verstöße zu erkennen und ihre Konsequenzen einzufordern.

Blog-Betreuung

Für die Betreuung des Blogs waren mehrere Projektmitglieder zuständig. Von den zuständigen Personen wurden Blog-Beiträge über den aktuellen Stand der Projektarbeit, Events und Workshops verfasst. Außerdem war die Blog-Betreuung dafür zuständig, alle Anlässe fotografisch festzuhalten.

2.3 Werkzeug-Auswahl

Für das Projektmanagement wurde auf diverse Tools zurückgegriffen. So wurde Atlassian Confluence als Wiki für den Wissensaustausch und die Ablage von Dokumenten rund um die Projektgruppe verwendet. Mit Jira war ein weiteres Atlassian Tool im Einsatz, welches als Ticket System gleichermaßen für die Umsetzung von Scrum als auch für die nachfolgende Kanban Methodik genutzt wurde. Für die Projektplanung wurde Microsoft Project verwendet. Hiermit wurde ab der zweiten Phase des Projekts die zeitliche Abfolge der Teilprojekte geplant und mittels eines Gantt Charts visualisiert.

2.4 Umsetzung

In diesem Unterkapitel wird die letztendliche Umsetzung in Bezug auf das Projektmanagement beschrieben. Begonnen wird mit dem eingesetzten Vorgehensmodell und im Anschluss wird auf die wöchentlichen Treffen, die Seminarphase sowie auf die Projektphasen und Meilensteine eingegangen.

2.4.1 Eingesetztes Vorgehensmodell

Die Projektgruppe hat sich in der ersten Phase für Scrum entschieden. Da die Bearbeitung der Seminarthemen aber die höchste Priorität hatte und auch den meisten Zeitaufwand einnahm, wurde von einer Aufteilung in einzelne Sprints abgesehen. So stellte die Seminarphase an sich einen längeren Sprint dar, in dem einzelne, einleitende Aufgaben, wie die Anforderungserhebung, Planung und diverse andere Dinge erledigt wurden. Nach der Seminarphase wurde ein Versuch unternommen, den ersten Sprint nach allen Scrum-Vorgaben umzusetzen. Da sich die Gruppe nur zweimal wöchentlich traf, war ein Daily Scrum nicht umsetzbar, so dass anstelle dessen ein Weekly Scrum eingeführt wurde.
Nach kurzer Zeit wurde allerdings klar, dass das Product Backlog noch zu klein war und der Scrum-Prozess an sich zu viel Overhead hatte, was somit eher den Projektfortschritt schmälerte. Gründe waren unter anderem der begrenzte zeitliche Umfang der Projektgruppe. Dies hatte zur Folge, dass die Projektgruppe sich für eine angepasste Form von Kanban entschieden hat. Bei dieser angepassten Form wurde das Weekly Scrum als Weekly Report

beibehalten und keine Priorisierung der Aufgaben durchgeführt, sodass die Aufgaben frei aus dem Aufgaben-Pool entnommen wurden durften. In den Meetings wurde dabei darauf geachtet, dass die wichtigsten Aufgaben erledigt wurden. Das Product Backlog durfte dabei zu jeder Zeit mit weiteren Aufgaben bestückt werden, was eine schnelle Reaktion auf Probleme ermöglichte. Die Ziele und somit auch die Aufgaben des Projektes wurden dabei anhand der Meilensteine abgeleitet. Dabei kam die Einsicht, dass die Zeiträume der definierten Meilensteine allerdings noch zu weit auseinanderlagen und kleine Zwischenziele formuliert werden sollten, um die Motivation für das Team zu erhöhen.

2.4.2 Wöchentliche Treffen

Verpflichtend für alle Projektmitglieder war die Teilnahme am wöchentlichen Meeting mit den Betreuern sowie den Ansprechpartnern seitens des Praxispartners. Die Meetings fanden in der Regel am Mittwoch statt und dauerten meistens nicht länger als eine Stunde. Es wurde im Rahmen des Weekly Reports von jedem Mitglied der Projektgruppe berichtet welche Tätigkeiten in der zurückliegenden Woche durchgeführt wurden und ob es Herausforderungen gegeben hat. Des Weiteren wurde Organisatorisches besprochen, wie beispielsweise die Terminierung der Zwischen- sowie Abschlusspräsentation. Die Ergebnisse jedes Meetings wurden protokolliert und das jeweilige Protokoll in Confluence bereitgestellt. Der Protokollant wechselte jede Woche in alphabetischer Reihenfolge, sodass jedes Projektgruppenmitglied Protokoll führen musste.
Des Weiteren fand in den ersten Monaten der Projektgruppe ein internes Meeting statt bei welchen nur die Mitglieder der Projektgruppe anwesend waren. Dieses Meeting wurde jedoch gegen Ende der ersten Projektphase aufgegeben, da sich hieraus kein größerer Mehrwert ergab. Wichtige Themen konnten im Anschluss an das wöchentliche Treffen geklärt werden oder über andere Kanäle wie bspw. WhatsApp.

2.5 Projektphasen und Meilensteine

Das Projekt wurde in vier Projektphasen aufgeteilt, die jeweils eine Dauer von drei Monaten hatten und zum Ende mit einer Zwischen- bzw. Abschlusspräsentation abgeschlossen wurden.

- Projektphase 1: 01.04.2018 - 29.06.2018 (01.07.)
- Projektphase 2: 01.07.2018 - 08.10.2018 (01.10.)
- Projektphase 3: (01.10.2018 - 01.01.2019)

- Projektphase 4: (01.01.2019 - 01.04.2019)

In der ersten Projektphase wurden anfangs die organisatorischen Fragen geklärt wie z. B. die Terminfestlegung für das interne Meeting als auch das wöchentliche Treffen. Des Weiteren fand von Anfang April bis Mitte Mai die Seminarphase statt, auf die im nächsten Kapitel näher eingegangen wird. Begonnen wurde zudem mit den Teilprojekten Brillenberater und Brillenpassleser, für die sich die Projektgruppe in zwei Teilgruppen gleicher Größe aufteilte. Das Vorgehensmodell während der ersten Projektphase war Scrum.
In der zweiten Projektphase wurde von Scrum auf Kanban umgestiegen. Des Weiteren wurde in Form eines klassischen Projektmanagements ein Gantt-Projektplan erstellt und für jedes Teilprojekt Arbeitspakete definiert. Es wurden mit der Social Media Trendanalyse und der Architektur/Infrastruktur zudem zwei weitere Teilprojekte gestartet und in den Planungsprozess mit aufgenommen. Auch die Zuordnung der Teilgruppen wurde geändert, sodass es nun Kompetenzteams gab, die sich auf ein Themengebiet wie beispielsweise Machine-Learning spezialisiert haben. Abgeschlossen wurde die zweite Phase am 8.10.2018 mit der Zwischenpräsenation in der alten Fleiwa in Oldenburg. Das Gantt-Diagramm, welches die Arbeitspakete für die Teilprojekte sowie die Meilensteine enthält, ist in Anhang 66 zu sehen.
Am Anfang der dritten Phase wurde das Teilprojekt Brillenpassleser abgeschlossen und mit Brillentinder ein neues Teilprojekt initiiert. Des Weiteren wurde ein Konzept zur Evaluation des Brillenberaters erarbeitet. Abgeschlossen wurde die dritte Projektphase mit der Präsentation des Zwischenstands am 21.12.2019.
In der vierten und letzten Projektphase wurden die Teilprojekte fertiggstellt sowie die Evaluation durchgeführt. Mit der Abschlusspräsentation am 09.04.2019, die wie die Zwischenpräsentation ebenfalls in der alten Fleiwa stattfand, endete die Projektgruppe. Der Projektplan für die dritte und vierte Phase ist in Anhang 67 zu finden.

2.6 Seminarphase

In der Seminarphase, die von Anfang April bis Mitte Mai andauerte, wurde von jedem Mitglied der Projektgruppe ein projektspezifisches Thema bearbeitet, um einen Einblick in die für das Projekt relevanten Technologien zu bekommen. Die Themen wurden durch die Betreuer vorgegeben und jedes Mitglied konnte sein Wunschthema angeben. Anschließend wurden die Themen entsprechend der Wünsche wie folgt verteilt:

- Eudes: Neuronale Netze vs. Machine Learning
- Robert: Convolutional Neural Networks - Feature Extractor

- Lea: Convolutional Neural Networks - Landmark Prediction and Regional Proposal
- Jonas: Recommender Systeme
- Simon: Frameworks für maschinelles Lernen (ausgestiegen, daher wurde das Thema nicht bearbeitet)
- Jannik: Best Practice - Betreiben und Warten von Big Data AI-Projekten
- Jannis: TensorFlow - Aufbau und Vorteile
- Felix: Business Intelligence - Integration von heterogenen Datenquellen
- Jan-Hendrik: Recurrent Neural Networks - Am Beispiel von Natural Language Processing
- Michael: Aktuelle Anwendungsfälle Artificial Intelligence
- Wiebke: Facial Feature Extraction
- Stefan: Neuro Fuzzy Systeme - Kombination von neuronalen Netzen und regelbasierten Systemen

Die Ausarbeitungen zu den Themen mussten zehn Seiten umfassen und bis zum 11.05.2018 abgegeben werden. Zudem musste zu jedem Thema ein 30 minütiger Workshop vorbereitet werden. Die Workshops wurden am 17. und 18.05.18 durchgeführt. Die Seminararbeiten sind in Anhang A.8 zu finden.

3 Entwicklungsumgebung

Im folgenden Kapitel wird in 3.1 die Werkzeugauswahl für den Entwicklungsworkflow beschrieben. 3.2 beschreibt den eigentlichen Entwicklungsworkflow.

3.1 Werkzeugauswahl

Die Auswahl der benötigten Werkzeuge erfolgte gemeinsam in der Gruppe und auf Basis der Erfahrung einzelner Gruppenmitglieder mit den ausgewählten Produkten. So wurden die Werkzeuge Jira, Bitbucket und Confluence ausgewählt, weil diese bereits von der Uni Oldenburg bereitgestellt wurden und hierdurch kein eigenes System aufgesetzt werden musste. Jenkins und Docker wurden ausgewählt, da einige Gruppenmitglieder bereits über Erfahrungen mit den Produkten verfügten und somit in der Lage waren diese zu administrieren. Für das Projekt wurden folgenden Werkzeuge ausgewählt:

Atlassian Jira Ticketsystem zur Verteilung von Aufgaben und Erfassung von Zeiten der Projektmitglieder.

Atlassian Bitbucket Quellcodeverwaltung zur Bereitstellung und Verwaltung von Code-Repositorys

Atlassian Confluence Wissensdatenbank (Wiki)

Jenkins Build-Umgebung zum Testen und Bereitstellen fertiggestellter Anwendungen/-Module

Docker Containervisualisierung für den Betrieb der Build- und Testumgebung.

Sonatype Nexus Artefaktserver für die Speicherung und Verwaltung von Entwicklungsartefakten

Zusätzliche Werkzeuge, die von den Projektmitgliedern für die Entwicklung genutzt wurden, werden an dieser Stelle nicht berücksichtigt.

3.2 Entwicklungsworkflow

Um zu verhindern, dass nicht getesteter, bzw. nicht lauffähiger Quellcode in die gemeinsame Codebasis gelangt wurde ein Entwicklungsworkflow eingeführt, welcher eine Überprüfung des Codes vor dem Einfügen in den Hauptzweig ermöglicht. Dieser Prozess wird von den in Abschnitt 3.1 beschriebenen Werkzeugen unterstützt. Vor der Beschreibung des Prozesses erfolgt ein kurzer Exkurs über das Versionierungssystem Git, welches

im Rahmen der Quellcodeverwaltungs- und Kollaborationsplattform Bitbucket verwendet wird.

Exkurs: Git Git ist ein Versionierungssystem, welches für die Versionskontrolle von Quellcode eingesetzt wird. Bei Git handelt es sich um ein verteiltes System, bei dem Entwickler auf einer lokalen Replikation des Quellcodes eines Projektes arbeiten. Dies ermöglicht es mehreren Nutzern zeitgleich zu arbeiten. Folgend werden die wichtigsten Begriffe erläutert [Tho19]:

Repository Beinhaltet alle Daten eines Projektes inklusive aller Versionen.

Commit Zusammenfassung von Änderungen an einer oder mehreren Dateien.

Branch Abzweigung eines Repositorys, welche nicht durch Änderungen in anderen Branches beeinflusst wird. I. d. R. verfügt jedes Repository über mindestens einen zentralen Hauptbranch, üblicherweise 'master' genannt.

Pull-Request Aufforderung einen abgezweigten Branch wieder mit dem Hauptbranch zusammenzuführen (mergen). Pull-Requests müssen von einem Teammitglied (Reviewer) geprüft werden bevor ein Merge erfolgt. Pull-Requests selber sind kein integraler Bestandteil von Git, sondern nutzen die bereits bestehenden Funktionalitäten von Branches und Merges.

Um zu verhindern, dass nicht getesteter bzw. nicht lauffähiger Quellcode in den Hauptbranch (in diesem Fall develop) gelangt, wurde der in Abbildung 1 dargestellte Prozess definiert:

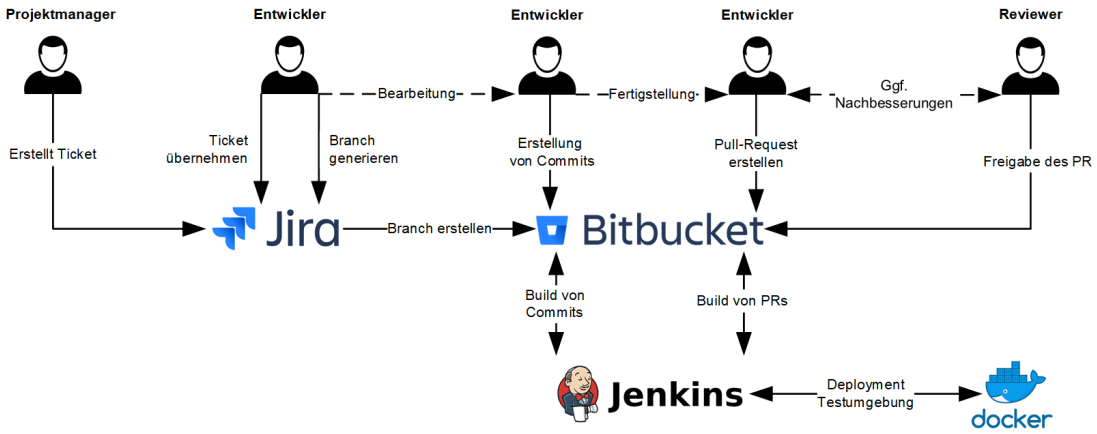


Abbildung 1: Entwicklungsworkflow

Der Prozess beginnt damit, dass der Projektmanager (in manchen Fällen auch ein Entwickler) eine Aufgabe als Ticket im Ticketsystem Jira hinterlegt. Ein freier Entwickler aus dem entsprechenden Teilbereich (z.B. Backend) übernimmt das Ticket und generiert über das Ticket in Jira einen neuen Branch in dem entsprechenden Repository (z. B. feature/PGPAI-343-doku-entwicklungsumgebung). Wird ein Branch direkt über ein Ticket erzeugt, wird dieser mit dem Ticket verknüpft. Dadurch kann Jira auf bestimmte Änderungen am Branch (z. B. Erstellung eines Pull-Requests) reagieren und z. B. den Status des dazugehörigen Tickets ändern. Nachdem der Branch erstellt wurde kann der Entwickler mit der Bearbeitung der Aufgabe beginnen und Änderungen in seinen Branch in Form von einem oder mehreren Commits einpflegen. Wird ein Commit durchgefühlt prüft das Build-System automatisiert, ob das Projekt noch lauffähig ist. Hierzu werden für jedes Projekt Build Pipelines in Form von sogenannten Jenkinsfiles gepflegt. Diese definierten, welche Schritte im Rahmen des Builds für ein Projekt durchgeführt werden. Typischerweise beinhaltet dies das Bauen von Docker Images, das Veröffentlichen dieser auf dem Artefaktserver des Projektes und das Anstoßen von weiteren Jobs auf dem Buildserver, welche dann wiederum die Testinstanz mit dem zuvor gebauten Artefakt aktualiseren.

Ist die Bearbeitung abgeschlossen erstellt der Entwickler einen Pull-Request - eine Aufforderung seinen Branch mit dem Basisbranch zusammenzuführen. Je nach Projekt wählt Bitbucket den Reviewer automatisch, der Entwickler kann diesen allerdings auch selbst festlegen. Der beauftragte Reviewer testet anschließend die vom Entwickler gemachten Änderungen und weist auf mögliche Fehler oder Verbesserungen hin, welche vom Entwickler behoben werden müssen. Wurden alle Anmerkungen bearbeitet kann der Reviewer den Pull-Request genehmigen und den Branch mergen, wobei letzteres nur möglich ist, wenn der Build auf dem Jenkins-Server erfolgreich war. Wird der Basisbranch durch einen Merge verändert beginnt Jenkins damit die Testumgebung automatisiert neu aufzubauen. Somit steht immer die aktuellste, lauffähige Version eines Projektes in der Testumgebung zur Verfügung.
Um die entwickelten Anwendungen und realen Bedingungen testen zu können werden diese auf Servern in der Amazon-Cloud (AWS) über Brille24 bereitgestellt und besteht aus drei Linux-basierten Instanzen auf denen die Anwendungen in Docker-Containern betrieben werden. Eine VM bildet dabei die Basis-VM, auf welcher alle Entwicklungsdienste betrieben werden, die nicht von der Uni Oldenburg betrieben werden. Dies umfasst den Jenkins-Buildserver und den Nexus-Artefaktserver. Auf der zweiten VM wird die permanent erreichbare Testinstanz betrieben. Die dritte VM wiederum ist der Build-Rechner, der vom Jenkins für die konkreten Builds der einzelnen Projekte verwendet wird. Die

Builds erfolgen nicht auf dem Jenkins-Master selber, damit fehlschlagende Builds oder Fehlkonfigurationen von Build-Jobs keinen negativen Einfluss auf die Erreichbarkeit der kritischen Entwicklungsdienste wie Nexus oder der Testinstanz haben.

4 Architektur

Der Themenkomplex Architektur beinhaltet neben Software- und Systemarchitektur der AI-Service-Plattform auch die Entwicklung geeigneter Vorgaben und eines Protokolls für eine einheitliche Struktur von Backend-Diensten, um den Aufwand für weitere zusätzliche Dienste gering zu halten. Im folgenden Kapitel wird der Themenkomplex beschrieben, die Anforderungen an die Themen, die in diesem Komplex bearbeitet werden, und welche Schritte im Rahmen der Umsetzung durchgeführt wurden dokumentiert. Abschließend erfolgt eine retrospektive Betrachtung der Arbeitsergebnisse und ein Abgleich, ob die Anforderungen umgesetzt werden konnten.

4.1 Motivation

Die Architektur einer Software hat immensen Einfluss auf die Wart-, Erweiterbar- und Skalierbarkeit von Softwaresystem. Insbesondere für verteilte Systeme ist eine korrekte Architektur unverzichtbar, da sich ohne diese der Aufwand für Änderungen mit jeder zusätzlichen Komponente im System stark erhöht. Um dies zu verhindern sollte eine Architektur gewählt werden, bei die einzelnen Dienste möglichst entkoppelt und unabhängig voneinander geplant, entwickelt und umgesetzt werden können.

Die Entwicklung dieser Architektur ist der zentrale Bestandteil dieses Themenkomplexes, jedoch nicht der einzige Teil. Neben der Architektur ist auch die Ausarbeitung eines allgemeinen Protokolls für die Kommunikation zwischen den Diensten der Plattform zu formulieren.

4.2 Anforderungen

Vorbedingung für die Umsetzung der in Abschnitt 4.1 aufgelisteten angedachten Eigenschaften für die Architektur ist die genaue Definition von Anforderungen an die einzelnen Teilaufgaben, die in diesem Themenkomplex bearbeitet werden sollen. Damit adäquate Arbeitsaufträge geschrieben werden können, muss klar definiert sein, welche funktionalen- und nicht-funktionalen Anforderungen das finale Artefakt erfüllen muss.

4.2.1 Protokoll

Das Protokoll, welches von den Diensten zur Kommunikation verwendet wird, besteht aus insgesamt fünf unterschiedlichen Nachrichtentypen. Die Anforderungen an dieses Protokoll werden im Folgenden definiert.

Die erste Anforderung an das Protokoll ist ein einheitliches Format um diese zu definieren. Aus diesem Grund wurde die Entscheidung getroffen, das Protokoll als JSON Schema zu definieren.

Exkurs: JSON Schema Das JSON Schema ist ein JavaScript Object Notation (JSON) Medientyp um die Struktur einer JSON-Datei zu definieren. Es dient zur Definition, Validierung, Dokumentation, Navigation und Interaktion von und mit JSON Daten [Int18]. Ein JSON Dokument ist eine Ressource, welche selbst durch den application/json Multipurpose Internet Mail Extensions (MIME)-Type beschrieben wird. JSON Schemata werden für JSON-Dokumente genutzt. Es können jedoch auch andere Dokumente oder Datenstrukturen, die auf ein JSON Schema-kompatibles Format abgebildet werden können, gegen ein solches Schema interpretiert werden. Bei Concise Binary Object Representation (CBOR), ein Datenformat mit dem Ziel möglichst wenige Code zu benötigten und gleichzeitig besonders kleine Nachrichten zu erzeugen, handelt es sich z. B. um eine solche als JSON abbildbare Datenstruktur.

Nr.	Beschreibung
ARC.1.1	**Offener Standard für alle Datenstrukturen** Alle Definitionen für Datenstrukuren, die im Rahmen der Architektur getroffen werden, müssen in einem offenen Standard wie z.B. JSON Schema oder OpenAPI Definitions ausformuliert werden.
ARC.1.2	**Minimale Parametermenge für externe Anfragen** Für alle Anfragen von externen Teilnehmern an Dienste in der Plattform ist ein Datenschmema zu definieren, welches eine Validierung eingehender Anfragen gegen eine minimale Parametermenge ermöglicht. Jede Anfrage muss dabei mindestens die Parameter *serviceName*, *apiVersion* und *requestPayload* enthalten. Zusätzlich muss der Parameter *modelVersion* im Schema mit einem entsprechenden Datentypen hinterlegt werden.

Tabelle 2: Anforderungen an das Protokoll - Teil 1

Nr.	Beschreibung
ARC.1.3	**Minimale Parametermenge für Antworten auf externe Anfragen** Für Antworten von Diensten auf externe Anfragen ist ein Datenschema zu definieren, welches eine maschinelle Validierung ermöglicht. Damit Rückschlüsse auf die Performance einzelner Dienste gezogen werden können, muss neben dem Antwort-Payload *responsePayload* auch eine eindeutige ID für den Request(*requestId*) enthalten sein. Daneben sollen auch folgende Parameter in allen Antworten enthalten sein: *serviceInstanceId*, *serviceName*, *startedAt*, *finishedAt*, *responseStatus*.
ARC.1.4	**Datenobjekt zur Serviceanmeldung** Dienste sollen sich an der Plattform anmelden können. Für diesen Vorgang muss ein Datenschema definiert werden, welches eine minimale Parametermenge definiert und wie bei den anderen Anforderungen auch eine maschinelle Validierung dieses Schemas ermöglicht. Die minimale Parametermenge besteht dabei aus *serviceName*, *apiVersion* und *instanceId*. Daneben sollen auch *instanceName*, *serviceVersion*, *modelVersion* und *revision* als optionale Argumente bei der Registrierung von Diensten an der Plattform erlaubt sein.
ARC.1.5	**Datenobjekt für den Beginn einer Anfragebearbeitung** Wenn mit der Bearbeitung einer Anfrage durch eine Dienstinstanz begonnen wird, soll eine interne Nachricht versendet werden. Für diese muss ebenfalls ein Schema existieren, welches die Datenstruktur definiert. Aus der Nachricht muss erkennbar sein, welche Instanz die Anfrage bearbeitet und wann mit der Bearbeitung begonnen wurde. Es müssen dabei minimal die Felder *serviceName*, *apiVersion*, *requestPayload*, *requestId*, *serviceInstanceId* und *startedAt* enthalten sein.

Tabelle 3: Anforderungen an das Protokoll - Teil 2

Das Listing 1 enthält das Schema für Dokumente, welche zur Anmeldung von Diensten an der Service Plattform genutzt verwendet werden können. Entscheidend ist dabei, dass bestimmte Werte wie der Name des Diensts (bzw. der Typ des Dienstes), die unterstützte Application-Programming-Interface (API)-Version und die eindeutige Kennung der Instanz Pflichtfelder sind und in jeder Anfrage vom Typ **ServiceRegistration** enthalten sein müssen. Dadurch wird sichergestellt, dass von Diensten, die solche Anfragetypen bearbeiten können, wie z. B. das Service Gateway (siehe Kapitel 7).

```
{
  "$id": "https://pg.joelt.de/schema/v1/service-registration.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "ServiceRegistration",
  "type": "object",
  "properties": {
    "instanceName": {
      "type": "string",
      "description": "Human readable name of the service instance. Will be auto-generated if not set."
    },
    "instanceId": {
      "type": "string",
      "description": "Unique id of the instance that will be included in debug information. Will be auto-generated if not set."
    },
    "serviceName": {
      "type": "string",
      "description": "The name of the service that is registered."
    },
    "apiVersion": {
      "type": "string",
      "description": "API Version of the registered service. E.g. v1 or v1-pre."
    },
    "serviceVersion": {
      "type": "string",
      "description": "Version of the service, e.g. 1.2.5"
    },
    "modelVersion": {
      "type": "string",
      "description": "Model version provided by this instance."
    },
    "revision": {
      "type": "string",
      "description": "Git revision of the service"
    }
  },
  "additionalProperties": false,
  "required": [
    "serviceName",
    "apiVersion",
    "instanceId"
  ]
}
```

Listing 1: Schema für Service Discovery Nachrichten (ServiceRegistration)

Neben diesen Pflichtangaben sind noch einige weitere optionale Werte in dem Anfragetyp vorgesehen. Diese sind für die ordnungsgemäße Funktion der Plattform nicht zwingend erforderlich, enthalten aber potentiell hilfreiche Informationen, wie ein für Menschen ver-ständlicher Name der Instanz, die Version des in der Instanz verwendeten Machine Learning Modells (falls vorhanden) und die Version des Services, inklusive Revision, um genau zurückverfolgen zu können, welcher Softwarestand bei der jeweiligen Dienstinstanz in Verwendung ist.

Feldname	**Typ**	**Beschreibung**
apiVersion	String	Die API Version gibt an, welche Funktionen eine bestimmte Instanz eines Dienstes mindestens bereitstellt. Die Version wird genutzt, um Anfragen an die entsprechend passenden Dienstinstanzen zur Bearbeitung zu leiten. Um eine größtmögliche Flexibilität zu ermöglichen, ist der Typ des Feldes String, da so die Nutzer der Plattform ein eigenes Schema für die API Version verwenden.
finishedAt	String	POSIX-Zeitstempel für den Beginn der Bearbeitung einer Anfrage. Diese wird von einem Dienst in dem Moment gesetzt, in dem mit der aktiven Bearbeitung einer Anfrage begonnen wird. Zeiten, in der die Anfragen in der Warteschlange verweilen, werden durch diesen Zeitstempel nicht erfasst. Kompatible Zeitstempel können durch folgenden Code erzeugt werden `str(datetime.utcnow().timestamp())`.
instanceId	String	Eindeutiges Identifikationsmerkmal für eine Instanz eines Dienstes. Das Feld dient der Zuordnung von Antworten zu Dienstinstanzen, um im Fehlerfall die verursachende Instanz zu identifizieren. Der Wert des Feldes ist i. d. R. eine UUID in String Darstellung.
instanceName	String	Verständlicher Name für eine Instanz eines Dienstes. Dient dazu Instanzen im Monitoring schnell zu identifizieren.

Tabelle 4: Übersicht der im Protokoll definierten Felder - Teil 1

Feldname	Typ	Beschreibung
modelVersion	String	Optionales Feld, welches dazu dient Dienstinstanzen getrennt von der Versionierung des Dienstes und der API das verwendete Machine Learning Modell zu versionieren. Wird wie serviceName und apiVersionm für das korrekte Routing von Anfragen verwendet.
requestId	String	Eindeutige Identifikationswert für eine Anfrage. Wird für das Routing von Antwortnachrichten verwendete und dient u. a. der Nachverfolgbarkeit von Nachrichten in der Plattform. I. d. R. wird für dieses Feld ein UUID in der String Darstellung verwendet. Die Nutzung von UUIDs ist nicht verpflichtend. Es kann grundsätzlich jeder nicht leere String verwendet werden.
requestPayload	Object	Inhalt einer Anfrage, die vom Dienst selber bearbeitet wird. Der Typ ist Objekt, d.h. dass ein Array oder ein Dictionary übergeben werden. Simple Datentypen wie String oder Float sind nicht erlaubt.
responsePayload	Object	Inhalt einer Antwort, die von einem Dienst durch die Bearbeitung einer Anfrage generiert wurde.
responseStatus	Enum	Gibt an, ob eine Anfrage erfolgreich bearbeitet werden konnte oder nicht. Erlaubte Werte sind ok und error.
revision	String	Optionale Quellcodeversion einer Dienstinstanz. Diese kann im Debugging verwendet werden, um genau zu identifizieren aus welchem Quellcodestand eine bestimmte Dienstinstanz erzeugt wurde, für den Fall, dass sie die serviceVersion trotz Codeänderungen nicht ändert (z. B. während der Entwicklung einer neuen Version).

Tabelle 5: Übersicht der im Protokoll definierten Felder - Teil 2

Feldname	Typ	Beschreibung
serviceInstanceId	String	Synonym zu instanceId
serviceName	String	Name, der einen Typ eines Dienstes definiert, z. B. Brillenberater. Dieser wird für das Routing von Nachrichten an die korrespondierenden Dienste genutzt.
serviceVersion	String	Version, die angibt, welchen Softwarestand eine Dienstinstanz hat. Inhaltlich orientiert sich die Version an folgendem Muster: 0.1 bzw. 0.1.0. Dieses Feld hat keinen Einfluss auf das Routing von Nachrichten und dient dem Monitoring und der Fehlersuche.
startedAt	String	POSIX-Zeitstempel, der von einer Dienstinstanz zu Beginn einer Anfragebearbeitung gesetzt wird. Dieser dient zusammen mit finishedAt dazu, um zu erfassen, wie lange Dienstinstanzen für die Bearbeitung von Anfragen brauchen, um Erkenntnisse darüber zu gewinnen, wie viele parallele Instanzen für einen bestimmten Dienst benötigt werden. Zeiten die Anfragen in der Warteschlangen verbringen werden in diese Bearbeitungszeiten nicht einbezogen.

Tabelle 6: Übersicht der im Protokoll definierten Felder - Teil 3

```
{
  "$id": "https://pg.joelt.de/schema/v1/service-process-internal.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "ServiceProcessInternal",
  "type": "object",
  "properties": {
    "serviceName": {
      "type": "string",
      "description": "The name of the service that is registered."
    },
    "apiVersion": {
      "type": "string",
      "description": "API Version required to answer this request."
    },
    "requestPayload": {
      "type": "object",
      "description": "Request payload that will be parsed by the backend service."
    },
    "requestId": {
      "type": "string",
      "description": "ID of the request for tracking purposes."
    },
    "modelVersion": {
      "type": "string",
      "description": "Model version to be used by the service."
    },
    "serviceInstanceId": {
      "type": "string",
      "description": "ID of the worker that processes the request."
    },
    "startedAt": {
      "type": "string",
      "description": "ISO timestamp of the processing begin."
    }
  },
  "required": [
    "serviceName",
    "apiVersion",
    "requestPayload",
    "requestId",
    "serviceInstanceId",
    "startedAt"
  ]
}
```

Listing 2: Schema für eine interne Bearbeitungsnachricht (ServiceProcessInternal)

4.2.2 Software- und Systemarchitektur

Der zweite große Bestandteil des Architektur-Bereichs ist die verwendete Software- und Systemarchitektur. Die Anforderungen für diese entstanden primär in einem Workshop mit den beteiligten Plattform-Entwicklern und Eingaben vom Projektpartner Brille24. Ziel der Architektur ist es, die Bereitstellung und die Weiterentwicklung der Plattform und deren Dienste möglichst einfach zu gestalten. Primäre Zielplattform für den späteren Betrieb wird eine Cloud-Umgebung sein, da der Projektpartner Brille24 bereits einen Teil seiner Infrastruktur über die Amazon Web Services in der Cloud betreibt. Eine vollständige Auflistung der Anforderungen an die Softwarearchitektur kann der Tabelle 7 entnommen werden.

Nr.	Beschreibung
ARC.2.1	**Dienstbasierte Systemarchitektur** Die Plattform soll aus mehreren eigenständigen Diensten bestehen. Dies soll es ermöglichen, einzelne Dienste unabhängig von anderen zu ändern, zu starten und zu beenden.
ARC.2.2	**Asynchrone nachrichtenbasierte Kommunikation** Die Kommunikation zwischen den Diensten soll durch den asynchronen Austausch von Nachrichtenobjekten erfolgen. Im Standardfall dürfen Dienste nicht blockieren, um auf Antworten zu warten. Wenn auf Nachrichten, ähnlich wie beim synchronen Nachrichtenversand, gewartet werden soll muss dies explizit angefordert werden.
ARC.2.3	**Verlässliche Nachrichten für aktive Dienste** Nachrichten, die versendet wurden, dürfen auch im Fehlerfall nicht verloren gehen. Insbesondere dann, wenn Dienstinstanzen neu gestartet werden oder temporär nicht erreichbar sind. Ein Sonderfall gilt, wenn keine Instanzen für einen Dienst verfügbar sind. Dann dürfen Nachrichten an diese nach einem konfigurierbaren Timeout verworfen werden.

Tabelle 7: Anforderungen an die Softwarearchitektur - Teil 1

Nr.	Beschreibung
ARC.2.4	**Horizontable Skalierbarkeit** Dienste müssen horizontal skalieren können. Das heißt, dass je nach Bedarf mehrere Instanzen eines Dienstes gestartet werden können, welche dann die eingehenden Anfragen lastverteilt abarbeiten.
ARC.2.5	**Wenig Plattformspezifika in Diensten** Der Quellcode eines Dienstes soll möglichst wenig Merkmale enthalten, die nur spezifisch für die Betriebsplattform sind. D.h. Dienste sollten möglichst nur den Quellcode enthalten, den sie für ihre angedachte Funktionalität benötigen. Die Menge des benötigten sogenannten Glue-Codes ist minimal zu halten.
ARC.2.6	**Einfache Erweiterung um neue Dienste** Die Plattform soll möglichst leicht durch neue Dienste und Diensttypen erweitert werden können. D.h. die Lernkurve für die Plattform ist flach zu halten.
ARC.2.7	**Alle Programmschnittstellen in Python** Die Schnittstellen, welche für die Anbindung weiterer Dienste an die Plattform zu verwenden sind, sind in der Programmiersprache Python bereitzustellen. Dies begründet sich darin, dass Python eine im Machine Learning-Bereich weit verbreitete Sprache ist und es auch für Data Scientisten, ohne Hintergrund in der Softwareentwicklung, möglich sein soll, neue Dienste für die Plattform zu entwickeln.

Tabelle 8: Anforderungen an die Softwarearchitektur - Teil 2

4.3 Umsetzung

Die Umsetzung der beiden Teilbereiche Protokoll sowie Software- und Systemarchitektur erfolgt, genau wie die Anforderungsaufnahme, getrennt. Im Folgenden ist dokumentiert, welche Schritte im Rahmen der Umsetzung der Anforderungen für die beiden Themenkomplexe durchgeführt wurden.

4.3.1 Protokoll

Die Umsetzungsschritte für das Protokoll bestanden primär aus der Auswahl eines geeigneten Standards zur Definition der Datenobjekte und die darauf folgende Ausformulierung der Anforderungen in der Grammatik des ausgewählten Standards. Da es für das

Protokoll keine Vorgaben oder Einschränkungen vom Projektpartner gab, wurde die Auswahl von den Entwicklern der Basisdienste in einer gemeinsamen Abstimmung vorgenommen. Die Wahl fiel auf JSON Schema, da sich mithilfe dieses weit verbreiteten Standards komplexe Objekte und Datenstrukturen definieren und validieren lassen, ohne das fixe Abhängigkeiten zu bestimmten Plattformen eingegangen werden müssen. Da externe Dienste über eine HTTP-Schnittstelle auf die Dienstplattform zugreifen und dort ohnehin JSON-Objekte ausgetauscht werden, bot sich das JSON Schema an.
Insgesamt wurden fünf JSON Schemata definiert. Das Listing 2 zeigt am Beispiel des internen Nachrichtenobjektes, welches zu Beginn einer Anfragenbearbeitung versendet wird, wie ein solches Schema aussieht. Bei den definierten Schemata bzw. Nachrichtentypen handelt es sich um die Folgenden:

- **ServiceRegistration:** wird versendet, wenn ein Dienst sich an der Plattform anmeldet und bereit zur Bearbeitung eingehender Nachrichten ist.
- **ServiceProcessInternal:** wird versendet, wenn ein Dienst mit der Bearbeitung einer Nachricht begonnen hat.
- **ServiceRequestInternal:** wird genutzt, um Anfragen an andere Dienstinstanzen und Typen innerhalb der Plattform zu versenden.
- **ServiceResponseInternal:** enthält die Antwort eines Dienstes auf eine interne Anfrage.
- **ServiceRequest:** wird von externen Nutzern versendet, um Anfragen an Dienste innerhalb der Plattform zu senden.

4.3.2 Software- und Systemarchitektur

Die Umsetzung der Anforderungen an die Software- und Systemarchitektur erfolgt primär in der Implementierung einer Basisbibliothek. Aufgrund des Umfanges der Basisbibliothek erfolgt die Dokumentation nicht an dieser Stelle, sondern in Kapitel 5.

4.4 Abnahme

Der Themenkomplex Architektur enthält eine Vielzahl von Anforderungen, die im Rahmen der Umsetzung berücksichtigt werden müssen. In diesem Abschnitt wird dokumentiert, welche diese Anforderungen während der Projektlaufzeit umgesetzt wurden. Wie auch die anderen Abschnitte in diesem Kapitel teilt sich die Abnahme in die beiden Teilbereiche Protokoll und Software- bzw. Systemarchitektur auf.

4.4.1 Protokoll

Durch die Definitionen der JSON-Schemata konnten alle Anforderungen an das Protokoll umgesetzt werden. In der Tabelle 9 sind die einzelnen Anforderungen und der jeweilige Aspekt, durch den diese umgesetzt wurden, aufgeführt.

ID	Titel	Status	Umsetzung
ARC.1.1	Offener Standard für alle Datenstrukturen	umgesetzt	Nutzung von JSON Schema für alle Definitionen
ARC.1.2	Minimale Parametermenge für externe Anfragen	umgesetzt	JSON Schemata: ServiceRequest & ServiceRequestInternal
ARC.1.3	Minimale Parametermenge für Antworten für externe Anfragen	umgesetzt	JSON Schema: ServiceResponseInternal
ARC.1.4	Datenobjekt zur Serviceanmeldung	umgesetzt	JSON Schema: ServiceRegistration
ARC.1.5	Datenobjekt für Beginn einer Anfragebearbeitung	umgesetzt	JSON Schema: ServiceProcessInternal

Tabelle 9: Abnahme der Protokoll-Anforderungen

4.4.2 Software- und Systemarchitektur

Da die Anforderungen an die Software- und Systemarchitektur im Rahmen der Entwicklung einer Basisbibliothek umgesetzt wurden, befindet sich die Abnahme dieser nicht in diesem Kapitel, sondern in Abschnitt 5.2 des Basisbibliothek Kapitels.

4.5 Fazit

Architektur als Themenkomplex stellt für die Entwicklung der Plattform und vor allem auch für die Erweiterbarkeit und Vielseitigkeit der Service Plattform einen integralen Bestandteil dar. Im Rahmen des Projektes konnten alle Anforderungen, die entweder durch den Projektpartner vorgegeben oder durch den die Projektteilnehmer selber in Workshops abgeleitet wurden, erfolgreich umgesetzt werden.
Der Arbeitsaufwand, der in den Themenkomplex Architektur investiert wurde, hat sich als im Verlaufe des Projektes als sinnvoll herausgestellt. Durch die Entwicklung der Basisbibliothek (siehe Kapitel 5) und des festen Protokolls konnte bei der Entwicklung neuer

zusätzlicher Dienste massiv Arbeitsaufwand eingespart werden.
Die Leistungsfähigkeit der Architektur hat sich in den durchgeführten Belastungstests wie erwartet als ausreichend herausgestellt, auch für den Betrieb im größeren Umfang, wie es z. B. bei einem zum Kunden gerichteten produktiven Einsatz der Fall wäre. Die genauen Ergebnisse der Belastungstests befinden sich im Kapitel 5, in dem auch die Basisbibliothek dokumentiert ist.

5 Basisbibliothek

Im Rahmen der Anforderungserhebung für die Software- und Systemarchitektur ist der Bedarf einer Softwarebibliothek entstanden, welche die fachliche Logik für die Kommunikation innerhalb der Service Plattform kapselt, damit bei der Entwicklung von zusätzlichen Diensten für die Plattform nur wenig fachliche Details der Plattform selber berücksichtigt werden müssen. Dienste sollen dann idealerweise nur den Quellcode enthalten, den sie für die Umsetzung der fachlichen Funktionalität benötigen.
Die genauen Anforderungen an die Basisbibliothek wurden bereits im Architektur-Kapitel in dem Abschnitt 4.2.2 definiert und werden aus diesem Grund nicht in diesem Kapitel wiederholt.

5.1 Umsetzung

Die Umsetzung der Anforderungen erfolgte durch die Implementierung einer Basisbibliothek in einer geeigneten Programmiersprache. Da für die übrigen Komponenten bereits die Sprache auf Python festgelegt und eine Anforderung existiert, welche die Schnittstellensprache vorgibt: ARC.2.7, Python für alle Schnittstellen, wurde, um keinen Bruch zwischen Diensten und der Basisbibliothek zu erzeugen, auch für die funktionale Implementierung die Sprache auf Python festgelegt.
Der nächste Schritt der Umsetzung bestand aus dem Anlegen einer geeigneten Projektstruktur. Python bietet für die Verteilung von Komponenten und Bibliotheken eine integrierte Paketverwaltung. Der Python Standard PEP-427 definiert ein einheitliches binäres Paketformat, welches für das Packen und Verteilen von Python-Bibliotheken genutzt werden kann [Hol12]. Python 3 enthält bereits im Standard das Paket setuptools, welches Hilfsmethoden für die Erstellung von Paketen bereitstellt. Dieses wird auch für das Basisbibliothek-Projekt verwendet. Im Anhang A.4 befindet sich die Projektdatei setup.py, in welcher definiert wird, welche Module ein Python Paket - in diesem Fall die Basisbibliothek - enthält.
Neben den Informationen, die für das Basisbibliothek-Paket benötigt werden, wie z. B. der Autorenname, die Abhängigkeiten zu anderen Paketen und Metadaten wie Name und Beschreibung, wird auch die Version in der Bibliothek gesetzt. Damit eine eindeutige Zuordnung zwischen den später ausgelieferten binären Pakten und einem Stand in dem dazugehörigen Git-Repository der Bibliothek möglich ist, wurden die beiden Methoden __extract_git_revision und __update_version eingeführt. Durch diese beiden Methoden wird beim Installieren des Paketes (was u. a. auf dem Buildsystem geschieht) die kurze

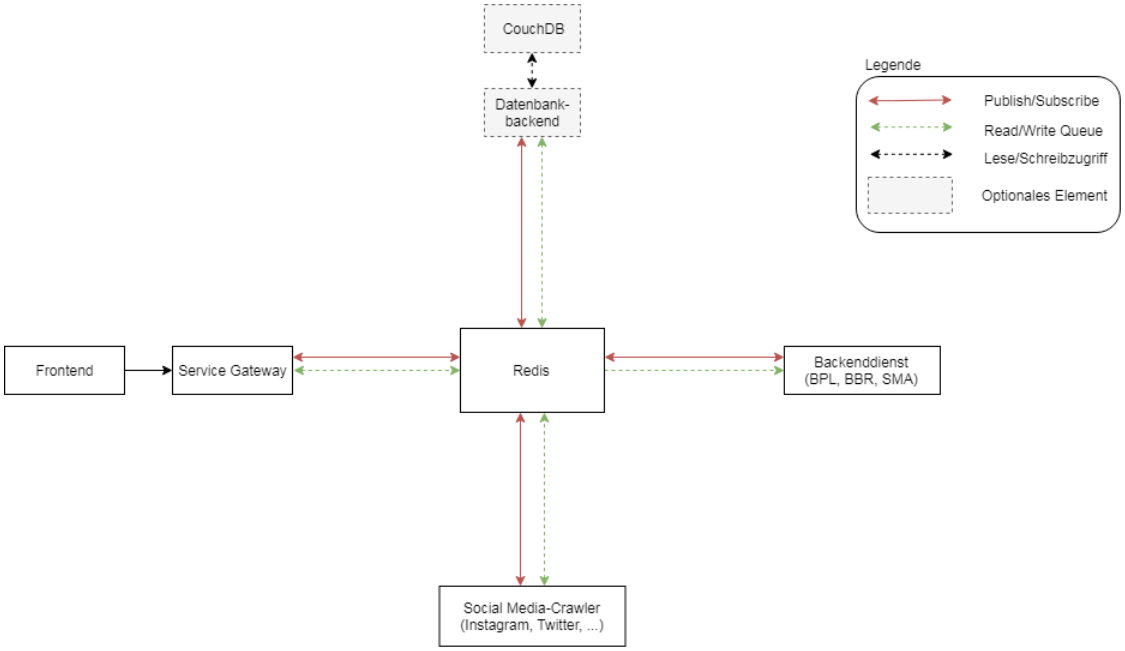


Abbildung 2: Nachrichtenbasierte Kommunikation zwischen Diensten

Variante der eindeutigen Git-Revision in eine **version.py** Datei innerhalb des paiservice Moduls geschrieben. Diese Datei wird mit in das Paket eingepackt und erlaubt, überall wo die Bibliothek eingebunden wird, das programmatische Abrufen der eindeutigen Version inklusive Git-Revision.

Eine weitere Anforderung, die im Rahmen der Basisbibliothek umgesetzt wurde, ist die nachrichtenbasierte Kommunikation zwischen Diensten und Dienstinstanzen (ARC.2.2). Zentraler Bestandteil dieser Kommunikation ist die Software Redis, ein verteilter Key/Value-Store, der u. a. auch die Möglichkeit bietet, per Publish/Subscribe Mechanismus Nachrichten zu verschicken. Die Abbildung 2 zeigt schematisch die Integration von Redis in die Plattform. Wie dort erkennbar, stellt Redis den zentralen Dreh- und Angelpunkt dar über den sämtliche Kommunikation abläuft.

Die Kommunikation erfolgt mittels Redis über das Anlegen von Keys auf dem Redis Server, wobei der Value immer ein JSON-Objekt ist. Auf die Kompression oder Serialisierung in weniger speicherintensive Binärformate wird an dieser Stelle verzichtet, um die Möglichkeit offen zu halten, die Nachrichten direkt auf dem Redis-Server einsehen zu können. Auch können so Webanwendungen, bei denen die Deserialisierung von Binärnachrichten nicht immer problemlos ist, auf den Redis Server zugreifen und so in die Plattform eingebunden werden. Dies dient der Erhaltung der Erweiterbarkeit der Plattform und ist nicht direkt

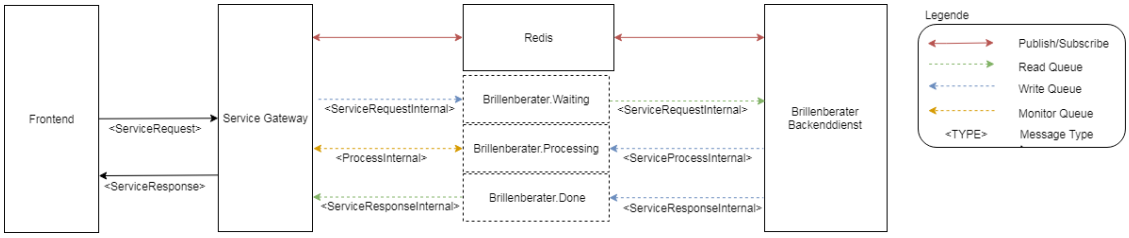


Abbildung 3: Nachrichten-Queues in der Servicekommunikation (Reliable-Queue-Pattern)

durch eine funktionale Anforderung begründet.
Die Kommunikation über Redis erfolgt dabei über Events und Key/Value-Einträge. Grundsätzlich wird bei jeder versendeten Nachricht zunächst ein Key auf dem Redis-Server angelegt, welcher repräsentiert, für welchen Dienst und welche API-Version die Nachricht bestimmt ist. Anschließend wird der Wert des Keys auf das Nachrichtenobjekt gesetzt, welches verschickt werden soll. Wie genau diese Objekte auszusehen haben, ist über das Protokoll (siehe Kapitel 4.2.1) definiert. Die Basisbibliothek enthält alle, im Rahmen des Protokolls definierten JSON-Schemata und validiert die Nachrichtenobjekte gegen diese, bevor Nachrichten über einen Redis-Server versendet werden.
Die Keys werden dabei anhand eines hierarchischen Schemas erzeugt, welches neben den Anfragetypen auch den Typs des angefragten Dienstes und die dazugehörige API-Version enthält. Der letzte Bestandteil ist dabei eine UUID, welche als eindeutiges Identifikationsmerkmal der Anfrage fungiert. pai:service:requests:type#{type}:api#{api_version}:event.type#{event_type}:id#{id}
Nachdem die Objekte erfolgreich auf dem Redis-Server angelegt wurden, wird ein Event in einem definiertem Channel ausgelöst. Andere Dienste oder Instanzen können bestimmte Channel abonnieren und so über neue Nachrichten benachrichtigt werden, ohne dass diese aktiv den Redis-Server abfragen müssen. Dies reduziert die Last auf dem Gesamtsystem. Das Schema nach dem die Namen der Channel gebildet werden, folgt dabei der Struktur, welche auch für die Keys verwendet wird.
Damit Nachrichten gepuffert werden können, wird der besondere Objekttyp List auf dem Redis-Server verwendet. Sobald ein Event für eine Nachricht ausgelöst wird, wird gleichzeitig die ID, welche auf das dazugehörige Objekt einer Nachricht zeigt, in einer Liste abgelegt. Es existiert für jeden Diensttypen und Nachrichtentypen eine eigene unabhängige Liste auf dem Server.
Damit bei diesem Prozess keine Nachrichten verloren gehen können, wenn z. B. ein Dienst unvorhergesehen heruntergefahren wird oder kurzzeitig nicht erreichbar ist, wird ein sogenanntes Reliable-Queue-Pattern verwendet. Dies ist ein Designpattern, welches vorgibt,

dass Nachrichten immer erst dann aus einer Liste bzw. Queue entfernt werden, wenn sie zu einer anderen hinzugefügt worden sind.
In der Basisbibliothek ist dies so implementiert, dass wenn ein Dienst eine Nachricht zur Bearbeitung aus der Warteschlange entnimmt, diese zeitgleich in eine Bearbeitungsliste hinzufügt wird. Redis bietet für diesen Vorgang das Kommando **RPOPLPUSH**. Dieses entfernt das letzte Element einer Liste und fügt es auf den Kopf einer anderen Liste in einer atomaren Operation ein. D. h. selbst wenn die Dienstinstanz, welche die Nachricht abgerufen hat während der Abfrage oder direkt danach abstürzt, geht die Nachricht nicht verloren, da sie bereits in einer anderen Liste bzw. Queue gesichert wurde. Die Abbildung 3 zeigt, welche Queues und Nachrichtenformate bei der Bearbeitung einer Anfrage durch einen Dienst in der Plattform involviert sind.
Die Plattform soll leicht um weitere Dienste erweitert werden können. Die Anforderung ARC.2.5 sieht vor, dass Dienste möglichst wenig plattformspezifische Eigenheiten implementieren müssen. Dies ist dadurch erreicht worden, dass die Basisbibliothek letztlich nur einen Callback vorsieht, der aufgerufen wird, sobald eine Anfrage eintrifft. Ein Dienst muss keinerlei Logik implementieren, um Anfragenbehandlungen durchzuführen. Auch für den Fall, dass die fachliche Logik in einem Dienst einen schwerwiegenden Fehler wirft, muss keine gesonderte Behandlung im Dienst erfolgen. Die Basisbibliothek implementiert eine rudimentäre Fehlerbehandlung für diese Fälle.
Damit sind die Fälle für eingehende Anfragen hinreichend beschrieben. Die Basisbibliothek soll jedoch auch für den Versand von Anfragen genutzt werden können. Für diesen Fall bietet die Library einmal die Variante des sogenannten „Fire and Forget“, bei dem eine interne Dienstanfrage versendet wird, ohne das auf eine Antwort gewartet wird. Dieser Modus bietet sich immer dann an, wenn es nicht wichtig ist, ob wirklich alle Anfragen erfolgreich bearbeitet werden oder andere fachliche Mechanismen implementiert sind, um die asynchrone Bearbeitung von Anfragen sicherzustellen.
Die grundsätzliche Kommunikation erfolgt immer asynchron. Da es jedoch auch vorkommen kann, dass Anfragen synchron bearbeitet werden müssen, vor allem wenn eine HTTP-Schnittstelle bedient werden soll, die auch im Service Gateway (siehe Kapitel 7) genutzt wird, bietet die Basisbibliothek die Funktion eine Anfrage zu senden und synchron auf eine Antwort zu warten. Dieser synchrone Aufruf blockiert den aufrufenden Thread bis eine Nachricht eingetroffen ist. Technisch ist dies so gelöst, dass ein Callback Manager innerhalb der Basisbibliothek auf eingehende Events auf bestimmten Channels wartet und daraufhin wieder einen Callback auslöst, welcher dann den blockierten Thread wieder freigibt.
Neben dem Handling von eingehenden und ausgehenden Nachrichten befindet sich auch die Logik für Heartbeats in der Basisbibliothek. Dieser Mechanismus wird verwendet,

damit sich Dienste, sobald sie sich für einen Anfragetypen registriert haben, regelmäßig am Redis-Server melden. Dadurch wird erreicht, dass ein Dienst, wie z. B. das Service Gateway, aktiv abfragen kann, ob für bestimmte Dienste überhaupt Instanzen vorhanden sind. Da Anfragen gepuffert werden, wenn keine Instanz bereit ist, um diese zu bearbeiten, kann dies durch reguläre Anfragen nicht festgestellt werden.
Ein weiteres Implementierungsdetail ist die Möglichkeit, den Prozess des Herunterfahrens von Diensten zu beeinflussen. In der Basisbibliothek selber wird diese Funktion dafür genutzt, Dienste vom Herunterfahren von der Plattform abzumelden und auch das Senden von Heartbeats zu stoppen. Dienste können ebendiesen Mechanismus benutzen um z. B. Modelle aus dem Speicher zu entfernen oder Datenbankverbindungen ordnungsgemäß herunterzufahren.
Der Mechanismus unterstützt dabei das „normale“ Herunterfahren, ausgelöst durch ein **SIGTERM** Signal an den Prozess (z. B. [STRG] + [C]) und das erzwungene Herunterfahren, ausgelöst durch ein weiteres **SIGTERM** Signal, wenn sich die Anwendung bereits im Prozess der Herunterfahrens befindet.
Beim regulären Shutdown werden alle Anfragen, die derzeit in Bearbeitung sind, noch abgearbeitet und auch die Heartbeats werden in diesem Zeitraum weiterhin versendet. Wenn alle ausstehenden Anfragen bearbeitet wurden, meldet sich der Dienst von der Plattform ab. Wird während dieses Zeitraumes erneut das Signal zum Herunterfahren gesendet, werden schlagartig alle laufenden Prozesse abgebrochen und die Anwendung beendet.

```
1 def respond_mode(shutdown_helper, service_helper):
2   print('Respond mode active...')
3   service_helper.discovery.register_service()
4   shutdown_helper.register_shutdown_hook(service_helper.discovery.
      unregister_service)
5   shutdown_helper.register_shutdown_hook(service_helper.
      unregister_blocking_request_handler)
6   shutdown_helper.register_shutdown_hook(service_helper.heartbeat.stop)
7   service_helper.register_blocking_request_handler(callback=
      __service_callback)
```

Listing 3: Shutdown-Manager am Beispiel des Dummy-Services

Das Listing 3 zeigt die Nutzung des Shutdown-Managers, wie er im Rahmen der Referenzimplementierung (siehe Kapitel 6) genutzt wird. Hier ist erkennbar, dass für die Registrierung eines weiteren Shutdown-Hooks eine einzige Zeile Code ausreicht. Die Reihenfolge, im dem die Shutdown-Hooks registriert werden, ist die gleiche Reihenfolge mit der sie während des Beendens ausgeführt werden.

5.2 Abnahme

In diesem Abschnitt erfolgt die Abnahme der Anforderungen an die Basisbibliothek. Die Anforderungen wurden im Abschnitt 4.2.2 des Architektur-Kapitels ausformuliert und werden hier nur zur besseren Übersicht erneut dargestellt. Alle Anforderungen, die an die Software- und Systemarchitektur gestellt wurden, konnten direkt oder indirekt durch die Implementierung der Basisbibliothek, deren Funktionalität in diesem Kapitel erläutert wird, umgesetzt werden. Der Tabelle 10 kann für jede Anforderung entnommen werden, durch welchen Aspekt der Basisbibliothek diese umgesetzt wurden.

ID	**Titel**	**Status**	**Umsetzung**
ARC.2.1	Dienstbasierte Systemarchitektur	umgesetzt	Basisbibliothek ermöglicht kleine leichte Dienste.
ARC.2.2	Asynchrone nachrichtenbasierte Kommunikation	umgesetzt	Redis-Server als zentraler Dreh- und Angelpunkt.
ARC.2.3	Verlässliche Nachrichten für aktive Dienste	umgesetzt	Implementierung des Reliable-Queue-Pattern in der Nachrichtenkommunikation.
ARC.2.4	Horizontale Skalierbarkeit	umgesetzt	Mehrere Instanzen eines Dienstes verteilen Last untereinander.
ARC.2.5	Wenig Plattformspezifikation in Diensten	umgesetzt	Einbettung nur über Python Abhängigkeit und einen Callback.
ARC.2.6	Einfache Erweiterung um neue Dienste	umgesetzt	Service muss nur einen Callback implementieren.
ARC.2.7	Alle Programmschnittstellen in Python	umgesetzt	Basisbibliothek als Python Paket (wheel).

Tabelle 10: Abnahme der Protokoll-Anforderungen

5.3 Belastungstests

Da sich für die Basisbibliothek und die Software- bzw. Systemarchitektur keine klassische Evaluation durch Nutzer anbot, wurde stattdessen ein Belastungstest durchgeführt, um sicherzustellen, dass die entwickelte Plattform auch für den produktiven Betrieb im größeren Maßstab geeignet ist.

Insbesondere der Aspekt der horizontalen Skalierung (Anforderung ARC.2.4) stand dabei im Fokus der Betrachtung. Für die Belastungstests wurde der vollständige Weg durch die Plattform getestet. Dies beinhaltet die Eingabe von Anfragen über das Service Gateway (siehe Kapitel 7), die Kommunikation über einen zentralen Redis-Server und die Beantwortung der Anfragen durch eine wechselnde Anzahl von Instanzen der Referenzimplementierung (siehe Kapitel 6).

5.3.1 Vorgehensweise

Für die Lasttests wurden vier Szenarien definiert, die sich aus jeweils einer anderen Anzahl von Anfragen, Instanzen oder Worker-Prozessen innerhalb einzelner Instanzen zusammensetzten. Ein wichtiges Kriterium ist, neben der reinen Zeit für die Beantwortung der Anfragen, die Feststellung, ob sich die Plattform im Aspekt des Leistunsgsbedarfs deterministisch verhält. Dies lässt sich u.a. dadurch erfassen, dass geprüft wird, wie groß die Varianz zwischen mehreren Durchläufen desselben Szenarios ist.
Zur Durchführung der Lasttests wurde eine virtualisierte Umgebung genutzt in welcher die Dienste in einem Kubernetes-Cluster bereitgestellt wurden. Die Dienste wurden dort mittels sogenannter Kubernetes-Manifeste gestartet. Diese Manifeste definieren, welcher Container mit welchen zusätzlichen Einstellungen gestartet werden soll. Auf eine vollständige Beschreibung von Kubernetes wird in dieser Stelle des Protokolls verzichtet. Die Funktionsweise von Kubernetes kann der Dokumentation des Projektes entnommen werden (siehe `https://kubernetes.io/`).
In jedem Szenario wurde nun ein Kubernetes-Job ausgeführt (siehe Listing 4), welcher über das Service-Gateway eine definierte Anzahl von Anfragen an die Referenzimplementierung sendet. In einigen Szenarien erfolgt das Versenden der Anfragen auch parallel in mehreren Threads. Hierfür wird das Programm GNU Parallel verwendet [Tan18].
Um die Zeit zu messen, welche für die Beantwortung einer definierten Anzahl von Anfragen benötigt wird, wurde das Kommandozeilenwerkzeug **time** verwendet. Dieses Tool erfasst insgesamt drei verschiedene Zeitwerte für die Ausführung eines Kommandos: *real*, die echt benötigte Zeit für die Ausführung des Kommandos und *user* und *sys*, jeweils die Prozessorzeit, die im User- bzw. Kernel-Modus verbracht werden für die Ausführung des Kommandos. Die beiden letzteren fallen i. d. R. wesentlich niedriger aus als *real*, da diese nicht die Zeiten beinhalten, die der Prozessor auf I/O-Operationen wartet oder gerade ein anderer Prozess durch den Scheduler ausgeführt wird.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: dummy-load-parallel
spec:
  template:
    spec:
      restartPolicy: Never
      containers:
      - name: dummy-load
        image: ubuntu
        command:
        - bash
        - -c
        - |
          apt-get update && apt-get install -yq curl parallel
          for j in `seq 1 10`; do
            echo "RUN $j"
            time for i in `seq 1 100`; do
              parallel -j 10 curl --silent --output /dev/null -XPOST http://pai-service-gw-service.pai/v1/request/dummy-service/v1 -d '{"serviceName": "dummy-service","apiVersion": "v1","requestPayload": {"requestType": "regular"}}'
            done
            echo ----------
          done
```

Listing 4: Job-Manifest für die Lasttests

5.3.2 Aufbau

Die Lasttests fanden in einer virtualisierten Umgebung statt.
Dem Hostsystem standen dabei für alle Tests folgende Ressourcen zur Verfügung:

- CPU: Intel Core i7 8700K (6x3,7Ghz, 12 HT-Cores)
- RAM: 32 GiB DDR4-2133
- Storage: Samsung 960 EVO NVMe SSD
- 2x 1Gbit Ethernet (via NIC Teaming)

Auf dem Hostsystem wiederum wurde eine virtuelle Maschine ausgeführt, welche folgende Eckdaten aufwies:

- Hypervisor: VMWare Workstation Pro 14.5
- CPU: 12 vCores

- CPU-Features: Intel VT-x/EPT, vIOMMU, performance counters
- RAM: 16 GiB
- NIC: 10 GBit VMWare NAT Adapter

Innerhalb der VM wiederum wurde das Tool **Minikube** eingesetzt um eine lokale Kubernetes Installation zu starten. Dieses Tool startet die Zielumgebung wiederum in einer VM. Um dies zu erreichen wurde der zusätzliche VM-Treiber KVM2 verwendet, welcher die Zielumgebung innerhalb einer Kernel-Virtual-Machine (KVM) über die QEMU Schnittstelle bereitstellt.
Diese zweite geschachtelte VM wies folgende Eckdaten auf:

- Hypervisor: KVM/ QEMU
- CPU: 8 vCores, Host-Passthrough (in diesem Fall von der ersten VM)
- RAM: 6 GiB
- NIC1: 10 GBit VirtIO Adapter (minikube-net)
- NIC2: 10 GBit VirtIO Adapter (NAT)

Die Lasttests fanden in dieser geschachtelt virtualisierten Umgebung statt, da dies auch die Umgebung ist, in der die Plattform entwickelt wurde. Das Hostsystem ist eine Windows 10 Workstation, welche wiederum primär virtuelle Maschinen für unterschiedliche Zwecke bereitstellt. Die ersten VM in der Kaskade ist die eine Fedora Workstation Entwicklungsmaschine, auf der alle benötigten Tools und Werkzeuge für die Entwicklung von Python-basierten Softwareprojekten installiert ist. Die letzte Ebene stellt die Minikube-VM dar. An dieser Stelle wird keine lokal in der Entwicklungsmaschine installierte Kubernetes Umgebung genutzt, um leichter sicherstellen zu können, dass alle getesteten Szenarien den gleichen Ausgangspunkt haben (einen leeren, neuen Kubernetes Cluster).

5.3.3 Szenarien

Im Rahmen der Lasttests wurden insgesamt vier verschiedene Szenarien getestet. Diese unterschieden sich, wie bereits angedeutet, durch die Anzahl der parallel angefragten Nachrichten, die Anzahl der zur Verfügung stehenden Nachrichtenprozessoren im Gateway und der Anzahl der beantwortenden Instanzen.

Szenario 1 Das erste Szenario stellt eine simple und minimalistische Installation der Plattform dar. Dabei existiert nur eine Instanz und die Anfragen kommen sequentiell.

- 1 Instanz des Service Gateways - 4 Worker-Prozesse
- 1 Instanz der Referenzimplementierung
- 10x 1000 Anfragen - sequentiell

Szenario 2 Das zweite Szenario erweitert das Erste um zusätzliche Worker-Prozesse und mehr antwortende Instanzen. Die Anfragen gegen die Plattform werden weiterhin sequentiell versendet.

- 1 Instanz des Service Gateways - 8 Worker-Prozesse
- 4 Instanzen der Referenzimplementierung
- 10x 1000 Anfragen - sequentiell

Szenario 3 Das dritte Szenario entspricht dem ersten Szenario nur, dass in diesem Fall die Anfragen an die Plattform nicht sequentiell sondern parallel versendet werden. Dies entspricht der Belastung, welche die Plattform bei Diensten ausgesetzt ist, die von Kunden bedient werden.

- 1 Instanz des Service Gateways - 4 Worker-Prozesse
- 1 Instanz der Referenzimplementierung
- 10x 1000 Anfragen - parallel (10)

Szenario 4 Das vierte Szenario entspricht dem Zweiten, inklusive der Änderungen des Dritten (parallele Anfragen).

- 1 Instanz des Service Gateways - 8 Worker-Prozesse
- 4 Instanz der Referenzimplementierung
- 10x 1000 Anfragen - parallel (10)

5.3.4 Ergebnisse

Die Lasttests schufen neue Einblicke und ermöglichen das Treffen von Aussagen über die zukünftigen Nutzungspotentiale der Plattform und auch ggf. über Schwachstellen in der Architektur.
Die Tabelle 11 enthält die Zeiten für die vier verschiedenen getesteten Szenarien. Die Dauer ist dort jeweils in Sekunden angegeben. Für jedes Szenario wurden insgesamt zehn identische Durchläufe durchgeführt, um sicherzugehen, dass die Ergebnisse nicht durch äußere Einflüsse verfälscht wurden.
Die Ergebnisse für die beiden Szenarien 1 und 2 stellen sich als sehr ähnlich heraus. Insbesondere Abbildung 4 zeigt die Zeiten für alle Szenarien. Deutlich wird dabei, dass die Szenarien 1 und 2 ähnlich sind und die 3 und 4 nahezu deckungsgleich.
Es wurden für alle Durchläufe Grafiken erzeugt, die dem Anhang A.5 entnommen werden können.
Die Real-Zeiten unterscheiden sich zwischen den Durchläufen. Insbesondere bei Erhöhung des Parallelisierungsgrades verringert sich die Zeit deutlich. Dies wird u.a. dadurch gezeigt,

Nr.	Szenario 1			Szenario 2		
	real	user	sys	real	user	sys
1	24.106	3.422	3.668	24.472	3.485	3.963
2	23.630	3.318	3.618	17.214	3.480	3.903
3	23.686	3.410	3.544	17.340	3.598	3.918
4	23.555	3.380	3.610	16.676	3.402	3.688
5	23.908	3.432	3.555	17.418	3.574	3.934
6	23.693	3.371	3.597	17.442	3.611	3.913
7	23.782	3.427	3.681	17.307	3.624	3.854
8	24.051	3.375	3.741	17.456	3.486	4.007
9	23.857	3.427	3.631	17.326	3.571	3.897
10	23.594	3.299	3.632	17.087	3.521	3.798

Nr.	Szenario 2			Szenario 4		
	real	user	sys	real	user	sys
1	6.239	5.081	1.229	6.156	5.033	1.201
2	6.620	5.350	1.367	6.224	5.073	1.169
3	6.266	5.088	1.251	6.063	5.006	1.143
4	6.113	4.980	1.220	6.039	4.872	1.240
5	6.213	5.141	1.167	6.089	4.961	1.201
6	6.291	5.099	1.268	6.027	4.908	1.210
7	6.393	5.263	1.241	6.133	5.121	1.106
8	6.619	5.368	1.342	6.045	4.951	1.174
9	6.283	5.107	1.259	6.107	4.991	1.180
10	6.216	5.093	1.187	6.086	4.970	1.195

Tabelle 11: Messergebnisse der Lasttests (in Sekunden)

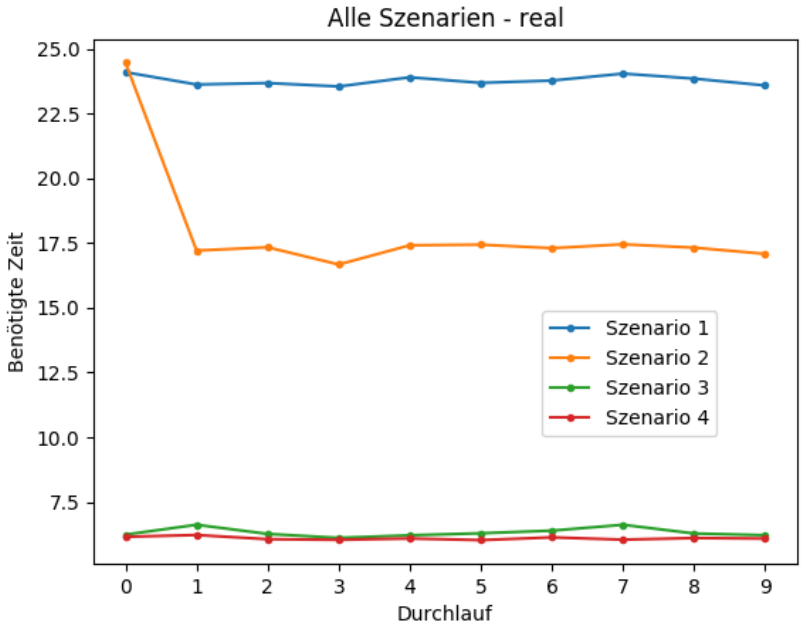


Abbildung 4: Real-Zeiten aller Szenarien

dass zwischen den ersten beiden und den letzten beiden Szenarien ein deutlicher Unterschied festzustellen ist. Dies begründet sich darin, dass bei der sequentiellen Abarbeitung der Anforderungen das Testsystem immer wieder auf die Antworten vom Service Gateway wartet, was die parallelen Tests jedoch auch tun, aber anders als die beiden letzteren nicht parallel mit der Vorbereitung für das Versenden der nächsten Anforderung arbeiten konnte. Dadurch erklärt sich die große Differenz der Real-Zeiten zwischen den einzelnen Szenarien.

Die Tests haben für uns bewiesen, dass das Konzept der horizontalen Skalierung im Rahmen der Plattform funktioniert. Durch die Bereitstellung weiterer Instanzen konnte die Reaktionszeit der Plattform bei großen Anfragemengen verbessert werden. Insbesondere im Vergleich von Szenario 1 zu Szenario 2 wird dies deutlich.

Ab einem gewissen Grad konnte durch die Lasttests keine zusätzliche Verbesserung der Geschwindigkeit gemessen werden. Dies betrifft die beiden Szenarien 3 und 4. Ab einem gewissen Punkt limitiert dort der Aufbau der Lasttests eine genauere Erfassung der Ergebnisse. Die Tests verbringen ab einem Punkt mehr Zeit damit, Prozesse zu starten und zu beenden, als mit der eigentlichen Bearbeitung der Anfragen. Zusätzlich zu diesen Einschränkungen wurden die Lasttests jeweils nur mit einer einzigen Instanz des Service

Gateways durchgeführt, wenn auch mit jeweils einer unterschiedlichen Anzahl von Worker-Prozessen. In einem produktiven Umfeld würde dieser Fall i. d. R. nicht vorkommen, da man dort einen Load-Balancer vor mehreren Instanzen des Gateways verwenden würde. Bei Einsatz eines solchen Aufbaus ist zu erwarten, dass die durchschnittliche Antwortzeit für weitere Anfragen weiter sinken wird.

5.4 Fazit

Die Basisbibliothek stellt den größten und umfangreichsten Aspekt der Software- und Systemarchitektur dar und steht somit stark im Fokus der Entwicklung. Alle Anforderungen, die im Rahmen des Projektes an die Architektur bzw. indirekt an die Basisbibliothek gestellt wurden, werden durch die Implementierung erfüllt.
Der Reduktion der Schnittstellen für Dienste, welche die Basisbibliothek nutzen, hat sich als sinnvoll erwiesen, da dadurch der Aufwand für die Implementierung zusätzlicher Dienste massiv verringert werden konnte.
Der Arbeitsaufwand, der in die Implementierung der Bibliothek geflossen ist, entsprach ungefähr den Erwartungen und hat so die Zeitplanung nicht negativ beeinflusst. Sowohl bei den Lasttests, als auch bei dem dauerhaften Betrieb der Testinstanzen während der Projektlaufzeit hat sich die Basisbibliothek als solide Software ohne großen Wartungsbedarf erwiesen. Die angedachten Konzepte der horizontalen Skalierung und der Implementierung des „Reliable Queue Patterns“ haben sich während der Projektlaufzeit als sinnvoll und technisch sauber umgesetzt herausgestellt.

5.5 Ausblick

Die Basisbibliothek kann sicherlich noch sinnvoll erweitert werden. So bleibt beim aktuellen Stand noch die Option offen, wie mit Anfragen umgegangen werden soll, deren Bearbeitung entweder nicht möglich oder unwahrscheinlich ist. Für diesen Zweck bieten sich z. B. Circuit-Breaker an, die bei der Bearbeitung von Anfragen hinterlegt werden könnten. Diese würden dann ggf. reduzierte und vereinfachte Ergebnisse zurück liefern anstatt Fehlermeldungen oder Timeouts.
Fehlerbehandlung ist in der Basisbibliothek bereits vorhanden, an dieser Stelle besteht jedoch auch noch Verbesserungspotenzial, insbesondere bei der Nachvollziehbarkeit von aufgetretenen Fehler. Hier wäre es z. B. sinnvoll den Zustand der Anwendung im Moment des Fehlers festzuhalten (CoreDump) und bei nicht-kritischen Fehlern die Ausgabe der Eingabeparameter auszuweiten.

6 Referenzimplementierung

Um zu gewährleisten, dass alle Services im Backend nach den gleichen Vorgaben aufgebaut werden, wird eine Referenzimplementierung für Services eingeführt. Diese beinhaltet ausschließlich die Basisbibliothek und implementiert die folgenden grundlegenden Funktionen:

- Ausgabe der Version
- Einlesen einer Konfigurationsdatei
- Interaktiver Servermodus
- Nicht-interaktiver Servermodus
- Nicht-interaktiver Clientmodus

Ausgabe der Version Die Version wird in der Konfigurationsdatei gepflegt. Zusätzlich wird eine Revisionsnummer ausgegeben. Hierbei handelt es sich um die Nummer des letzten Git-Commits. Dies ermöglicht eine genau Unterscheidung zwischen verschiedenen Versionen und Revisionen.

Einlesen einer Konfigurationsdatei Die Konfigurationsdatei enthält Servicetyp, Version, API-Version, sowie die Redis-Konfiguration (Hostname und Port). Die Datei entspricht dem YAML-Format und wird durch einen Helper aus der Basisbibliothek eingelesen und bereitgestellt.

Interaktiver Servermodus Der interaktive Servermodus wurde zu Testzwecken implementiert und wird in keinem produktiven Service verwendet. Er ermöglicht die Versendung verschiedener Nachrichten per Tastendruck. Es können fehlerhafte, normale oder große Requests versendet werden, welche mehr Bearbeitungszeit in Anspruch nehmen als normale Requests. Der Modus wurde hauptsächlich zu Beginn des Projektes genutzt, um die Kommunikation zu testen.

Nicht-interaktiver Servermodus Der nicht-interaktive Servermodus sendet jede Sekunde einen normalen Request und kann ebenfalls zu Testzwecken genutzt werden. Zusätzlich dient er als Referenzimplementierung für das Versenden von Requests. Listing 5 zeigt die Referenzimplemtierung für den nicht-interaktiven Servermodus.

```
1 def request_mode(service_helper, shutdown_helper):
2   shutdown_helper.register_shutdown_hook(hard_shutdown_for_server)
3   print('Request mode active...')
4   while True and not shutdown:
```

```
5     request = ServiceRequestInternal(request_payload={"requestType": "
      regular"})
6     print('Sending regular request - ', request.request_id)
7     service_helper.send_new_request_internal(request=request)
8     time.sleep(1)
```

Listing 5: Referenzimplementierung zum Versenden von Anfragen

Nicht-interaktiver Clientmodus Das Empfangen von Anfragen wird im nicht-interaktiven Clientmodus implementiert. Dieser registriert eine Callback-Methode, die bei einem eingehenden Request ausgeführt wird. Listing 6 zeigt die Implementierung für den Clientmodus.

```
1  def respond_mode(shutdown_helper, service_helper):
2    print('Respond mode active...')
3    service_helper.discovery.register_service()
4    shutdown_helper.register_shutdown_hook(service_helper.discovery.
       unregister_service)
5    shutdown_helper.register_shutdown_hook(service_helper.
       unregister_blocking_request_handler)
6    shutdown_helper.register_shutdown_hook(service_helper.heartbeat.stop)
7    service_helper.register_blocking_request_handler(callback=
       __service_callback)
8
9
10 def __service_callback(body):
11   print("Processing request body: ", body)
12   if body.get('requestType') is not None and body['requestType'] == '
       hard_error':
13     raise Warning('Unrecoverable error during processing')
14   if body.get('requestType') is not None and body['requestType'] == '
       soft_error':
15     return False, InternalProcessingError(message='Recoverable error during
       processing')
16   if body.get('requestType') is not None and body['requestType'] == 'slow':
17     time.sleep(5)
18   return True, {"foo": "bar"}
```

Listing 6: Referenzimplementierung zum Empfangen von Anfragen

Die Referenzimplementierung bietet somit eines Basis für alle Services im Backend. Für die einzelnen Services werden dann die benötigten Python-Pakete installiert und die Client bzw. Servermethode entsprechend angepasst. Die Kommunikation erfolgt, wie in der Referenzimplementierung, über die Basisbibliothek.

7 Service Gateway

Dieses Kapitel beschreibt die Entstehung des Service-Gateways. Hierzu wird in 7.1 die Grundidee des Teilprojektes beschrieben. Anschließend werden in 7.2 die Anforderungen und in 7.3 deren Umsetzung beschrieben. Abschließend erfolgt in Abschnitt 7.4 die Abnahme der Anforderungen und in 7.5 und 7.6 wird ein Fazit gezogen und ein Ausblick über weitere Möglichkeiten gegeben.

7.1 Beschreibung

Das Service Gateway ist eine Anwendung, welche Zugriff auf die internen Dienste der Service Plattform für externe Nutzer bieten soll. Für den externen Zugriff stellt das Gateway eine HTTP API bereit und setzt die ankommenden Anfragen in asynchrone Nachrichten um, die dann per Redis an die jeweiligen Backend-Dienste weitergeleitet werden bzw. abgeholt werden können. Das Service Gateway bildet dabei den einzigen Zugriffsweg für Dienste und Nutzer, die nicht selber Bestandteil der Service Plattform sind. Dadurch ist es möglich an dieser Stelle eine zentrale Authentifizierung und Autorisierung zu implementieren. Backend-Dienste können dadurch auf eine eigene Nutzerverwaltung und Mechanismen zur Zutrittskontrolle verzichten. Dies spart Entwicklungsaufwände ein, gibt dadurch aber vor, dass die Umgebung, in welcher die Service Plattform ausgeführt wird, vertrauenswürdig sein muss, um die Integrität und Sicherheit der Plattform als ganzes zu gewährleisten.

7.2 Anforderungen

Folgend werden die Anforderungen für das Service Gateway definiert. Die Anforderungen wurden teilweise durch den Projektpartner Brille24 vorgegeben. Andere Anforderungen wurden in einem gemeinsamen Workshop erarbeitet oder entstanden während der Entwicklung. Die jeweilige Herkunft wird jeweils direkt in der Anforderungsdefinition beschrieben.

Nr.	Beschreibung
SGW.1.1	**Bereitstellung einer HTTP-API** Zur Kommunikation mit anfragenden Diensten außerhalb der Plattform wird ein API-Endpunkt benötigt. Dieser soll mittels HTTP kommunizieren und in der Lage sein mehrere Anfragen gleichzeitig abzuarbeiten. Es muss möglich sein Endpunkte für HTTP-POST und HTTP-GET-Anfragen bereitzustellen. **Herkunft:** Vorgegeben von Brille24
SGW.1.2	**Anbindung an den zentralen Redis-Server** Zur Kommunikation mit Diensten innerhalb der Plattform wird eine Anbindung zum zentralen Redis-Server benötigt. Das Gateway muss in der Lage sein Anfragen für einen Dienst in die Redis-Datenbank zu schreiben und diese nach Bearbeitung durch einen Backend-Dienst zu lesen. **Herkunft:** Vorgegeben von Brille24
SGW.1.3	**Überwachung von Requests** Requests sollen während ihrer Bearbeitung durch einen Backend-Dienst überwacht werden, um den Status der Aufgabe nachzuverfolgen. Hierdurch soll verhindert werden, dass eine Anfrage nicht beantwortet wird, weil die Bearbeitung im Backend fehlschlägt. **Herkunft:** Definiert im Workshop zum gesamten Aufbau der Architektur
SGW.1.4	**Überwachung von Instanzen im Backend** Backend-Dienste müssen periodisch überwacht werden um sicherzustellen, dass diese Verfügbar sind. Ist für einen vom Gateway angebotenen Dienst keine Instanz im Backend verfügbar, muss die Anfrage abgelehnt werden. **Herkunft:** Definiert im Workshop zum gesamten Aufbau der Architektur

Tabelle 12: Anforderungen an das Service-Gateway - Teil 1

Nr.	Beschreibung
SGW.1.5	**Bereitstellung eines API-Endpunktes zur Abfrage von Metriken** Für die Überwachung des Gateways soll ein Endpunkt bereitgestellt werden über den Metriken abgefragt werden können. Es sollen folgende Metriken bereitgestellt werden: • Insgesamt verarbeitete Anfragen • Insgesamt fehlerhafte Anfragen • Insgesamt offene Anfragen • Durchschnittliche Antwortzeit aller Services • Durchschnittliche Antwortzeit je Service • Registrierte Instanzen • Offene Anfragen je Service • Abgeschlossene Anfragen je Service • Fehlerhafte Anfragen je Service **Herkunft:** Während der Entwicklung ursprünglich zu Test- und Debugzwecken implementiert und später als Anforderungen aufgenommen, um ein späteres Monitoring zu ermöglichen.
SGW.1.6	**Authentifizierung für externe Dienste mittels OAuth2.0** Um zu gewährleisten, dass kein externer Dienst ohne Genehmigung einen Dienst abfragen kann, soll eine Authentifikation mittels OAuth2.0 erfolgen. **Herkunft:** Definiert im Workshop zum gesamten Aufbau der Architektur

Tabelle 13: Anforderungen an das Service-Gateway - Teil 2

7.3 Umsetzung

Im Folgenden wird beschrieben, wie die in 7.2 definierten Anforderungen umgesetzt wurden.

SGW.1.1 Bereitstellung einer HTTP-API Die Bereitstellung der HTTP-Endpunkte erfolgt durch das Python-Paket *aiohttp*. Anders als das bekannte Paket *Flask* unterstützt *aiohttp* die asynchrone Kommunikation mittels HTTP. Das bedeutet, dass der Prozess im Service Gateway nicht durch eine Anfrage blockiert wird, weil auf die Antwort gewartet wird. Somit kann das Service Gateway zeitgleich mehrere Anfragen bearbeiten.

Nach dem Eingang einer Anfrage wird das ServiceRequest-Object in ein Service-Request-

Internal-Object übersetzt, welches auf dem zentralen Redis-Server abgelegt wird. Nach der Verarbeitung wird ein ServiceResponse-Object mit dem Ergebnis der Verarbeitung zurück an den anfragenden Client gesendet.

SGW.1.2 Anbindung an den zentralen Redis-Server Die Anbindung an den zentralen Redis-Server erfolgt durch die in Kapitel 4 beschriebenen Basis-Bibliothek. Diese wird im Service Gateway importiert.

SGW.1.3 Überwachung von Requests Da *aiohttp* eingehende Anfragen asynchron verarbeitet, somit also nicht auf einen Request wartet, müssen eingehende Requests überwacht werden. Hierzu ist in der Basis-Bibliothek die Methode *send_request_internal_with_response* implementiert. Diese sendet einen Request an den zentralen Redis-Server und registriert einen Callback-Worker für den Request, welcher überwacht, ob dieser abschließend bearbeitet oder fehlgeschlagen ist. Anschließend wird das Ergebnis an *aiohttp* übergeben und an den anfragenden Client, als Antwort auf seine Anfrage, übermittelt.

SGW.1.4 Überwachung von Instanzen im Backend Beim Start einer neuen Instanz registriert diese sich automatisch am Redis-Server. Hierüber überwacht das Service-Gateway, wie viele Instanzen für einen angebotenen Dienst zur Verfügung stehen. Steht keine Instanz eines angefragten Service zur Verfügung, wird der Request durch das Gateway mit *No Instance found* beantwortet.

Um sicherzustellen, dass keine Anfrage angenommen wird, wenn für den angefragten Service zwar eine Instanz registriert ist, diese allerdings durch ein aufgetretenes Problem nicht zur Verfügung steht, senden alle Instanzen zyklisch (alle 10 Sekunden) einen Heartbeat an den Redis-Server. Dieser Heartbeat zeigt an, dass der Service wie erwartet zur Verfügung steht. Das Service-Gateway prüft ebenfalls zyklisch, ob für jeden Dienst ein Heartbeat eingegangen ist. Bleibt dieser sechs mal (60 Sekunden) aus, wird die Registrierung des betroffenen Service entfernt und dieser erhält keine neuen Anfragen.

SGW.1.5 Bereitstellung eines API-Endpunktes zur Abfrage von Metriken Für das Abfragen der Metriken werden HTTP-GET-Endpunkte bereitgestellt
(z.B. */v1/stats/requests/all*). Diese liefern die Antwort im JSON-Format zurück. Die Metriken werden im Service-Gateway erfasst und im Redis-Server hinterlegt. Von dort ruft die Methode des Endpunkts die Daten ab und liefert diese an ein für diesen Zweck entwickeltes Dashboard.

SGW.1.6 Authentifizierung für externe Dienste mittels OAuth2.0 Bei OAuth2.0 handelt es sich um ein im RFC6749 beschriebenes Framework zur Authentifizierung von HTTP-Services. Das Protokoll definiert hierbei drei Rollen. Der *Resource Owner* erlaubt einem Client den Zugang zum *Authorization Server*, also die generelle Möglichkeit den Zugang zu einer geschützen Ressource anzufragen. Der *Authorization Server* wickelt die eigentliche Authentifizierung ab, indem er dem anfragenden Client einen Access Token ausstellt, mit dem er die geschützte Ressource vom *Resource Server* anfragen kann. [Dic12]

Das Service Gateway stellt in diesem Fall den *Resource Server* dar von dem eine Ressource (der Zugang zu einem Dienst) angefragt wird. Der *Authorization Server* wurde testweise mittels AWS Cognito (Vorgabe Brille24) realisiert. Hierbei kam es allerdings zu Problemen mit den ausgestellten Token und der Vergabe von Berechtigungen für die einzelnen Dienste. Aus diesem Grund wurde die Implementierung von OAuth nicht weiter verfolgt. Eine spätere Implementierung im Service Gateway wäre allerdings technisch möglich.

7.4 Abnahme

Folgend wird die Erfüllung der einzelnen Anforderungen bewertet. Erfüllte Anforderungen werden mit **Abgenommen**, nicht erfüllte Anforderungen mit **nicht Abgenommen** markiert. Anforderungen, die auf andere Art und Weise gelöst wurden erhalten die Markierung **nicht umgesetzt**

Nr.	Beschreibung	Abgenommen
SGW.1.1	Bereitstellung einer HTTP-API	**Abgenommen**
SGW.1.2	Anbindung an den zentralen Redis-Server	**Abgenommen**
SGW.1.3	Überwachung von Requests	**Abgenommen**
SGW.1.4	Überwachung von Instanzen im Backend	**Abgenommen**
SGW.1.5	Bereitstellung eines API-Endpunktes zur Abfrage von Metriken	**Abgenommen**
SGW.1.6	Authentifizierung für externe Dienste mittels OAuth2.0	**nicht umgesetzt**

Tabelle 14: Abnahme der Anforderungen an das Service-Gateway

7.5 Fazit

Bis auf die Authentifizierung für externe Dienste mittels OAuth2.0 wurden alle Anforderungen umgesetzt. Das Service Gateway bildet hiermit eine zentrale Rolle in der Gesamtarchitektur und stellte sich während des Projektes als sehr zuverlässig dar.

7.6 Ausblick

Mögliche Weiterentwicklungspotentiale im Hinblick auf einen möglichen Produktiveinsatz bietet das Gateway vor allem im Bereich Monitoring. Hier wäre es denkbar einen eigenen Monitoring-Agenten zu implementieren, welcher Daten über die bisher gesammelten Daten hinaus bietet. Außerdem sollte die Implementierung einer Authentifizierungslösung weiter verfolgt werden.

8 Datenbankservice

Um gesammelte Daten speichern zu können, wird ein Datenbanksystem benötigt. Während des Architekturworkshops wurde aus diesem Grund entschieden einen Datenbankservice in die Plattform aufzunehmen. Dieser besteht aus einer Datenbank und einem Schnittstellendienst, der zur Kommunikation mit anderen Dienstinstanzen vorgesehen ist.

8.1 Beschreibung

Als Datenbanksystem wurde CouchDB ausgewählt. Bei CouchDB handelt es sich um eine dokumentenorientierte Datenbank, welche den Datenbestand in JSON-Dokumenten verwaltet. Da die gesamte Kommunikation in der Service-Architektur auf JSON-Objekten basiert können diese somit ohne einen zusätzlichen Konvertierungsschritt gespeichert werden.

Der CouchDB-Instanz wird ein Datenbankdienst vorgelagert, welcher Anfragen empfängt und an die Datenbank sendet. Dieser Aufbau hat den Vorteil, dass keine spezifische Logik in die einzelnen Services integriert werden muss und bei einem Austausch des Datenbanksystems nur der Datenbankservice angepasst werden muss. Die fachliche Logik zur Interaktion mit der CouchDB Datenbank befindet sich somit vollständig im Datenbankdienst selber. Für die Nutzer des Dienstes ist nicht ersichtlich und auch nicht von Relevanz welche Datenbank in der Persistenzschicht verwendet wird.

8.2 Anforderungen

Folgend werden die Anforderungen definiert. Teilweise wurden diese durch den Projektpartner Brille24 vorgegeben, andere in einem gemeinsamen Workshop erarbeitet oder entstanden während der Entwicklung. Die jeweilige Herkunft wird jeweils direkt in der Anforderungsdefinition beschrieben.

Nr.	Beschreibung
DBS.1.1	**CRUD-Operationen** Der Service ist in der Lage CRUD (Create, Read, Update, Delete)-Anfragen durchzuführen. **Herkunft:** Erarbeitet im Workshop
DBS.1.2	**Suche nach ID-Präfixen** Der Dienst unterstützt die Suche nach Objekten anhand eines Präfixes auf die Dokumenten-ID (z. B. crawler-). **Herkunft:** Erarbeitet im Workshop
DBS.1.3	**Anfragen mittels ID** Eine Suche anhand einer eindeutigen Dokumenten-ID erfolgen. Der Datenbankdienst gibt genau das Dokument zurück, welches durch die ID identifiziert wird. **Herkunft:** Erarbeitet im Workshop
DBS.1.4	**Speicherung von Bildern als Anhang** Bilder werden nicht direkt am Datensatz gespeichert, sondern als Anhang zu dem jeweiligen Datensatz. **Herkunft:** Erarbeitet im Workshop
DBS.1.5	**Abrufen von Bildern als Anhang** Als Anhang gespeicherte Bilder können aus der Datenbank abgerufen werden. **Herkunft:** Erarbeitet im Workshop

Tabelle 15: Anforderungen an den Datenbankservice

8.3 Umsetzung

Um die Verbindung zur Datenbank herzustellen und einfache CRUD-Operationen zu implementieren wird das Pip-Paket *couchDB* eingesetzt (DBS.1.1). Die Suche funktioniert mittels sogenannter Mango-Querys. Mango ist eine in JSON deklarierte Abfragesprache [Cou19]. Die Pyhton CouchDB-Libary bildet alle einfachen CRUD-Operationen inkl. der Suche nach genauen IDs ab (DBS.1.1 und DBS.1.3). Die Suche nach ID-Präfixen (DBS.1.2) wurde mittels Mango-Query realisiert und wäre prinzipiell über die vorhandene Suchmethode möglich. Um die Suche allerdings zu vereinfachen, wurde ein weitere Methode implementiert, welche die benötigte Query bereits beinhaltet. Listing 7 zeigt die Funktion inklusive der benötigten Mango-Query:

```
def search_by_prefix(self, prefix, limit):
  """
  Searches for documents based on a prefix on the document id.

  :param prefix: Prefix to search for
  :param limit: Maximum of number of response objects

  :return: Query result
  """
  mango_query = json.loads('''{
    "selector": {
    "_id": {
      "$regex": "^''' + prefix + '''.*"
      }
    },
    "limit": ''' + str(limit) + '''
  }''')
  return self.search(mango_query)
```

Listing 7: Methode zur Prefixsuche

Eine Abfrage dieser Art würde dann wie folgt aussehen:

```
request = ServiceRequestInternal(request_payload={
  'requestType': 'prefix_search',
  'prefix': 'crawler-query-',
  'limit': 200
})
```

Listing 8: Abfrage zur Prefixsuche

Listing 8 zeigt eine Prefix-Suche für alle Datensätze mit dem Prefix *crawler-query-*. Hierzu wird der Typ *prefix_search* genutzt.
Da alle Bilder zwischen den Diensten mittels Base64 kodiert und versendet werden, erfolgt die Speicherung ebenfalls als Base64-String. Um die versendete Datenmenge zu reduzieren werden Bilder als Base64-Anhang an die Dokumente gehängt. Bei der Abfrage eines Datensatzes wird nicht der Anhang übertragen sondern lediglich die Dateinamen der verfügbaren Anhänge. Das anfragende System muss in einer separaten Anfrage den Anhang anfragen, um diesen zu bekommen.

```
1  def get_attachment(self, document_id, filename):
2    attachment_bytes = self.db.get_attachment(document_id, filename)
3    if not attachment_bytes:
4    return False, AttachmentNotFoundError(document_id, filename)
5
6    attachment_b64 = base64.b64encode(
7    attachment_bytes.read()).decode('ascii')
8    return True, {"b64": attachment_b64}
9
```

Listing 9: Methode zur Abfrage von Anhängen

```
1  get_attachment_payload = {
2    'documentId': document_id,
3    'requestType': 'get_attachment',
4    'filename': 'test-file.txt'}
5
```

Listing 10: Abfrage von Anhängen

Listing 9 zeigt die Methode zum Abruf von Anhängen, Listing 10 die dazugehörige Abfrage an den Dienst.
Die Verwendung von Anhängen hat den Nachteil, dass zusätzliche Nachrichten versendet werden müssen, allerdings gleichzeitig den Vorteil, dass keine unnötig großen Datensätze versendet werden. Insbesondere bei (Such)Anfragen mit großen Ergebnismengen kann so die Menge der zu übertragenen Daten massiv reduziert werden. Dies gilt besonders dann, wenn das anfragende System keine Verwendung für die Bilddaten hat, da nur die Metadaten benötigt werden (DBS.1.4 und DBS.1.5).

8.4 Abnahme

Folgend wird die Erfüllung der einzelnen Anforderungen bewertet. Erfüllte Anforderungen werden mit **Abgenommen**, nicht erfüllte Anforderungen mit **nicht Abgenommen** markiert. Anforderungen, die auf andere Art und Weise gelöst wurden erhalten die Markierung **nicht umgesetzt**.

Nr.	Beschreibung	Abgenommen
DBS.1.1	CRUD-Operationen	**Abgenommen**
DBS.1.2	Suche nach ID-Prefixes	**Abgenommen**
DBS.1.3	Anfragen mittels ID	**Abgenommen**
DBS.1.4	Speicherung von Bildern als Anhang	**Abgenommen**
DBS.1.5	Abrufen von Bildern als Anhang	**Abgenommen**

Tabelle 16: Abnahme der Anforderungen an den Datenbankservice

8.5 Fazit

Alle Anforderungen konnten erfüllt werden und der Datenbankdienst bietet zusätzlich die gewünschte Kapselung der Datenbank. In den meisten Services muss keine zusätzliche Logik für den Zugriff auf die Datenbank implementiert werden, lediglich in einigen Sonderfällen werden spezifische Mango-Querys in anderen Services genutzt.

8.6 Ausblick

Zukünftig könnten zur Verbesserung der Kapselung alle außerhalb des Datenbankservice genutzten Mango-Querys als Funktionen in den Datenbankservice integriert werden. Dies hätte den Vorteil, dass eine vollständige Kapselung erreicht werden kann. Nachteil hierbei ist allerdings, dass hierbei spezifische Anwendungslogik aus den einzelnen Services in den Datenbankservice wandern würde und somit die Komplexität steigen würde. Eine weitere Erweiterungsmöglichkeit wäre die Anbindung an das BI-System des Projektpartners Brille24, damit die gesammelten Daten für Analysen genutzt werden können, ohne dass eine Anpassung aller Dienste benötigt wird.

9 Brillenberater

Der Brillenberater stellt einen der zentralen Dienste dar, die im Verlauf der Projektgruppe konzipiert und ausgearbeitet wurden. Im Kern stellt der Brillenberater einen intelligenten Service dar, der potentiellen Kunden beim Einkauf von Brillen im Onlinehandel unterstützen soll. Auf Basis des abfotografierten Gesichtes des Kunden werden ausgewählte Machine Learning Algorithmen angewendet, um spefizische Features des Gesichtes zu extrahieren. Diese Features dienen als Grundlage, um letztendlich auf den Kunden zugeschnittene Brillenvorschläge zu generieren. Bevor tiefer in den Brillenberater eingestiegen werden kann, soll zunächst die zentrale Problemstellung dargelegt werden, die mithilfe des Dienstes des Brillenberaters adressiert werden soll.

9.1 Beschreibung

Um einen Einstieg in die Problematik des Brillenhandels im E-Commerce zu erhalten, ist ein Grundverständnis des Brillenhandels im Allgemeinen erforderlich. Es muss die Frage beantwortet werden, auf welche Art und Weise Brillen heutzutage gekauft werden und über welche Channel der Kunde potentielle Händler erreichen kann. Im Kern kann der Brillenhandel in zwei unterschiedliche Channel getrennt werden, dem Brillenhandel im E-Commerce mit E-Shops als Anlaufstelle und den lokalen Handel in stationären Geschäften. Beide Optionen weisen wesentliche Unterschiede auf und können mithilfe drei wesentlicher Merkmale charakterisiert werden: Sortiment, Beratung und Preis. Im Folgenden sollen diese Unterschiede kurz veranschaulicht werden. Angefangen mit dem Merkmal des Sortiments lässt sich feststellen, dass deutliche Unterschiede in der angebotenen Sortimentsbreite sowie -tiefe im E-Commerce und lokalen Handel existieren. Im lokalen Handel ist das angebotene Sortiment auf die zur Verfügung stehende Verkaufsfläche begrenzt. Im E-Commerce existieren diese Limitationen nicht, je nach Kapazität der zur Verfügung stehenden Ressourcen kann das Produktsortiment beliebig skalieren, sowohl in der Breite als auch in der Tiefe. Neben der größeren Produktauswahl existieren ebenfalls Preisvorteile beim Kauf in einem ausgewählten Online-Shop. Ein Online-Shop kann aufgrund von Kostenvorteilen Produkte meist günstiger anbieten als der Konkurrent im lokalen Handel. Unter Anbetracht der zuvor vermittelten Informationen sollte man meinen, dass wenn sowohl das Sortiment als auch der Preis des Onlinehandels dem im lokalen Geschäft überlegen ist, der Onlineshop in jedem Falle die attraktivere Alternative darstellt. Unter Berücksichtigung des dritten Merkmals, der persönlichen Beratung, kann sich diese Einstellung je nach Betrachtung der Kundengruppe jedoch drastisch ändern. Die

persönliche Beratung des stationären Handels ist das Alleinstellungsmerkmal und der zentrale Vorteil des lokalen Handels gegenüber dem E-Commerce. Kunden haben im Falle des stationären Handels stets die Möglichkeit, auf eine persönliche Beratung durch einen Mitarbeiter zurückzugreifen. Vor allem in Anbetracht modischer Aspekte ist dies ein zentraler Vorteil, da ein Kunde mitunter unschlüssig sein kann, ob jeweiliges Design des betrachteten Modells zu ihnen passt, was die aktuellen Modetrends sind und dergleichen. Aufgrund von Erfahrung, Intuition und Domänenwissen kann der lokale Mitarbeiter oft abwägen, ob eine Brille zu einem Kunden passt und innerhalb kürzester Zeit kundenspezifische Produktvorschläge übermitteln.
Dieser Vorgang kann im Kern durch einen abstrahierten Prozess verdeutlicht werden. Im Wesentlichen erfolgt eine Filterung des verfügbaren Sortiments. Auf Basis extrahierter Gesichtsmerkmale kann durch Gewichtung der zuvor gewonnenen Informationen der bestehende Pool an Produkten eingeschränkt werden. Dieser Filterungsprozess resultiert in einem kundenbezogenen Pool an Produktvorschlägen, die dem Kunden anschließend unterbreitet werden können. Es ist zu beachten, dass nicht alle Kunden Interesse an einem solchen Service haben, jedoch besteht stets die Möglichkeit, auf diesen zurückzugreifen. Bei Inanspruchnahme ist dem Kunden somit versichert, auf unkomplizierte Art und Weise die passende Brille für sich zu finden.
Zwar existieren bereits diverse Ansätze, um eine solche Beratung auf den E-Commerce zu übertragen, allerdings verlangen diese stets eine Interaktion mit dem Kunden. welche sich im Durchklicken durch unterschiedliche Features und Kriterien widerspiegelt. Die Abstinenz einer persönlichen Beratung im E-Commerce ist umso schmerzvoller in Anbetracht des wesentlich größeren Sortiments. Für den Kunden wäre es vom zentralen Vorteil, wenn er auf ähnliche Filterungsprozesse wie im stationären Handel zurückgreifen könnte. Bei Inanspruchnahme würde dies in einer deutlichen Reduzierung im Suchaufwand resultieren und zugleich direkt zu einem potentiellen Ergebnis führen.

An dieser Stelle setzt der Dienst des Brillenberaters an. Ziel des Brillenberaters ist es, die Beratung des stationären Mitarbeiters in den E-Commerce zu übertragen. Im Kern geht es darum, die zielgerichtete Filterung des Sortiments durch den Mitarbeiter auf Basis extrahierter Features in der E-Commerce-Domäne zu simulieren. Grundsätzlich soll dieser Filterungsprozess auf Basis eines Bildes des Gesichtes des jeweiligen Kunden passieren, welches er innerhalb des zur Verfügung gestellten Web-Services übermitteln kann. Anhand des Kundenbildes sollen anschließend unterschiedliche Klassifizierungen durchgeführt werden. Mithilfe der Klassifizierungen werden Features extrahiert, welche letztendlich ausschlaggebend für die individuellen Produktvorschläge sind. Bereits zu Beginn

wurde im Kontext des Brillenberaters von einem „intelligenten“ Dienst gesprochen, an dieser Stelle soll tiefergehend auf diesen Begriff Bezug genommen werden. Der Dienst wird als intelligenter Dienst bezeichnet, da er automatisiert kontextsensitive Klassifizierungsentscheidungen in einer Reihe disjunkter Prozesse treffen können muss. Als Grundlage dazu dienen verschiedene Machine Learning Modelle, welche die Basis für die angesprochene Extraktion jeweiliger Features darstellen. Im Kontext künstlicher Intelligenz und Deep Learning existieren unterschiedliche Themenbereiche, der Bereich Computer Vision sticht im Rahmen der Projektgruppe jedoch besonders hervor, da dieser genau mit den Anwendungsfällen unserer Klassifizierungsaufgaben tangiert. Als Grundlage dienen Convolutional Neural Networks, welche sich als State-of-the-Art Lösung für Klassifizierungen auf Basis von Bilddateien herausgestellt haben. Wesentlicher Meilenstein dafür stellte der damalige Sieg des Convolutional Neural Networks AlexNet im Rahmen der ImageNet-Competition dar, welches der damaligen Konkurrenz deutlich überlegen war. ImageNet ist eine Datenbank mit mehr als 14 Millionen annotierten Bilddateien. 2012 war das erste Jahr, in welchem ein Projekt auf Basis von Deep Learning den Wettbewerb gewann und in einem enormen Popularitätsschub von Deep Learning resultierte [Rus+15b].

9.2 Vorgehen

Ziel des Brillenberaters ist es, auf Basis von Klassifizierungen spezifische Merkmale eines Gesichts als Features zu extrahieren, diese anschließend entsprechend zu gewichten und auf Basis dessen kundenbezogene Produktvorschläge zu generieren. Zunächst musste die Frage beantwortet werden, welche Merkmale und Features für den Brillenberater überhaupt von Relevanz sind. Als erster Arbeitsschritt fiel also an, jeweilige Features zu identifizieren und abzuwägen, ob sie relevant für eine modische Beratung sind oder nicht. Dies geschah unter enger Zusammenarbeit mit dem Praxispartner Brille24, welcher dem Projektteam Fashion- und Designexperten zur Verfügung stellte, um einen Überblick zu Design- und Fashionaspekten bei der Brillenberatung zu vermitteln. Auf Basis dieser gewonnenen Informationen konnten verschiedene Arbeitspakete definiert werden, welche im Folgenden aufgeführt werden. Anforderungen ML:

- Klassifizierung Geschlecht
- Klassifizierung Alter
- Klassifizierung Hautfarbe
- Klassifizierung Bart/Nicht-Bart

- Klassifizierung Gesichtsform

Wie an der Aufteilung der Klassifizierungsaufgaben zu erkennen ist, erfolgt die Extraktion der Features nach einem Divide and Conquer-Prinzip. Anstelle der Klassifizierung sämtlicher Merkmale innerhalb eines einzelnen Algorithmus, wird das Gesamtproblem in verschiedene Teilprojekte zerlegt und jede Klassifizierung einzeln betrachtet. Wie bereits erwähnt dienen Convolutional Neural Networks als Grundlage zur Bearbeitung oben genannter Klassifizierungen. Auf den generellen Aufbau und deren Funktionsweise soll an dieser Stelle jedoch nicht weiter eingegangen werden. Im Anhang dieses Dokuments ist eine Seminararbeit zum Thema Convolutional Neural Networks hinterlegt, die die elementaren Grundlagen vermittelt. Die eigentliche Wahl der Netzwerkarchitektur stellt bei der Bearbeitung der aufgeführten Arbeitspakete allerdings einen Teilprozess dar, der im Gesamtprozess eher am Ende angesiedelt ist. Zur erfolgreichen Bearbeitung einer solchen Klassifizierungsaufgabe wird vor allem eins benötigt, Daten. Daten sind die Grundlage eines jeden Deep Learning Projekts und passende Datenquellen zu finden, ist nicht immer ganz trivial. Zunächst muss ein entsprechendes Domänenwissen aufgebaut werden, um passende Datenquellen zu identifizieren. Diese Datenquellen müssen möglicherweise anschließend noch transformiert werden und auf den jeweiligen Anwendungsfall angepasst werden. Hinzu kommt noch der Faktor, dass die akquirierten Daten möglicherweise nicht die nötige Qualität aufweisen und dementsprechend Qualitätsverbesserungen vorgenommen werden müssen. Bereits hier lassen sich diverse Muster erkennen.

Die zuvor genannten Arbeitsschritte fallen, wenn auch teilweise in abgewandelter Form, in jedem Machine Learning Projekt an. Um einen allgemeinen Leitfaden zum Bearbeiten der anfallenden Arbeitspakete im Bereich Machine Learning zu haben, wurde auf ein bekanntes Vorgehensmodell zurückgegriffen, dem CRISP-DM-Zyklus [Rie12].

CRISP-DM ist im eigentlichen Sinne ein Standard-Vorgehensmodell zum Data Mining. Obwohl es sich bei den definierten Arbeitspaketen im Bereich Machine Learning zwar nicht um Data-Mining-Tasks handelt, kann der Prozess dennoch auf die gesetzten Klassifizierungsaufgaben übertragen werden. Es existieren viele Gemeinsamkeiten beim Vorgehen in Data-Mining- sowie Machine-Learning-Anwendungsfällen. Vor allem im Prozess des Domänen-, des Datenverständnis und der Datenvorbereitung lassen sich die Muster auf den jeweiligen Anwendungsfall problemlos projizieren. Der CRISP-DM-Zyklus besteht aus insgesamt sechs Phasen. Im Folgenden sollen die einzelnen Phasen des CRISP-DM-Zyklus kurz erläutert werden und anschließend auf den jeweiligen Anwendungsfall unseres

Projektes übertragen werden. Es wird gezeigt, wie die jeweiligen Phasen sich innerhalb der Arbeitspakete unseres Projektes wiederspiegeln und welche Bedeutung diese für uns hat. Am Ende soll ein grundlegender Überblick über die Vorgehensmuster geschaffen werden, um diese anschließend auf die jeweiligen Klassifizierungsaufgaben zu übertragen

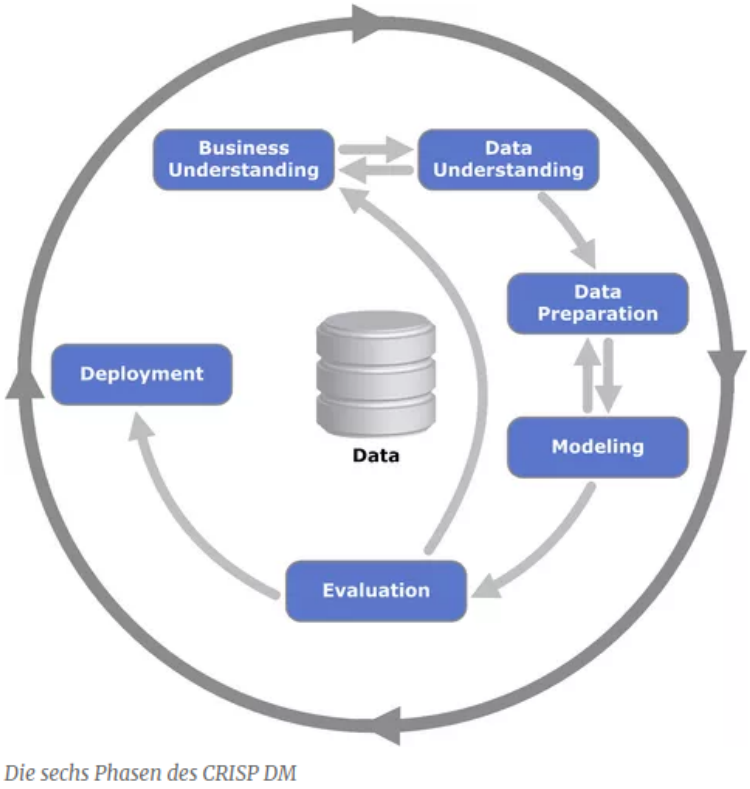


Abbildung 5: Die sechs Phasen des CRISP DM [Rie12]

1. Business Understanding

   Das Business Understanding repräsentiert den ersten Schritt im CRISP-DM-Zyklus und stellt die Grundlage eines jeweiligen Anwendungsfalls dar. Im Vordergrund steht die Aneignung eines Geschäftsverständnisses um letztendlich anhand des Domänenwissens konkrete Ziele und Anforderungen festzulegen und zu definieren. Durch das ableiten und der Definition von Aufgabestellungen soll ein zielgerichtetes Vorgehen im Projekt ermöglicht werden. Die Phase bietet das Fundament, um ausgehend davon das Projekt weiter auszubauen. Bereits in der Beschreibung des Brillenberaters wurde ein allgemeines Geschäftsverständnis unseres jeweiligen Anwendungsfalles verdeutlicht.

Durch die Identifikation der Problemstellung, der Aneignung von Domänenwissen wie den Unterschieden zwischen On- und Offlinehandel von Brillen, der konkreten Zieldefinition und der Ableitung von Anforderungen an den Dienst des Brillenberaters konnte eine Grundlage geschaffen werden, um ausgehend dieser immer spezifischer werdende Arbeitspakete zu definieren.

2. Data Understanding

Auf Basis des gewonnenen Geschäftsverständnisses muss weiterführend ein Datenverständnis geschaffen werden. Sowohl Data Mining- als auch Machine Learning-Anwendungsfälle sind extrem datengetrieben. Es ist also unabdingbar, zusätzlich ein spezifisches Domänenwissen im Bereich der Daten bezüglich des jeweiligen Anwendungsfalles aufzubauen. Daten stellen die Grundlagen eines jeden Machine-Learning-Projekts und sind ausschlaggebend für deren Erfolg oder Misserfolg. Im Kern geht es darum, ausgehend vom betrachteten Anwendungsfall, passende Datenquellen zu identifizieren, die Daten zu akquirieren und dahingehend zu transformieren, dass sie auf den Anwendungsfall bestmöglich zugeschnitten sind. Doch zur Transformation der Daten später mehr. Bezogen auf unsere Anwendungsfälle wird beispielsweise deutlich, dass wir vor allem Datensätze benötigen, auf denen Personen abgebildet sind. Bezogen auf unseren Anwendungsfall kann folgendes Beispiel zum verdeutlichen des Begriffs Datenverständnis genannt werden. Betrachtet man den Anwendungsfall der Klassifizierung des Alters einer übergebenen Person, kann die Schlussfolgerung gezogen werden, dass vor allem Datensätze mit unterschiedlichen Altersgruppen benötigt werden. Diese Erkenntnis kann den Aufwand bei der Suche passender Datenquellen deutlich reduzieren. Daten treten allerdings in unterschiedlicher Form auf. Um ein wenig Hintergrundwissen zu schaffen: Wie bereits erwähnt stellen Daten die Grundlage eines Machine-Learning Projektes dar. Gewonnene Daten werden in Form eines sogenannten Trainingsets zusammengefasst. Im Internet existieren viele unterschiedliche Open-Source-Datensätze, welche beispielsweise von Forschungsgruppen oder anderen Institutionen bereitgestellt werden. Für die Anwendungsfälle in unserem Projekt konnte auch auf eben genannte Open-Source-Datensätze zurückgegriffen werden, wie zum Beispiel das Datenset „Labeled Faces in the Wild“ [Uni18].

Vorgefertigte Datensätze haben den Vorteil, dass viele Vorbereitungsmaßnahmen, um die Daten für das Training vorzubereiten, bereits erledigt sind. Da die Datensätze bereits annotiert sind, müssen zudem keine zeitintensiven Labelaufgaben am jeweiligen Datensatz mehr durchgeführt werden. Für einige Anwendungsfälle existieren allerdings noch keine vorgefertigten Datensets und dementsprechend muss-

ten im Rahmen der Projektgruppe von Grund auf neue Datensets erstellt werden, die den Anforderungen des jeweiligen Anwendungsfalles entsprechen. Hierzu genügt es nicht, lediglich zu identifizieren, welche Daten benötigt werden. Nach der Akquirierung mussten zusätzlich diverse Verarbeitungs- und Transformationsprozesse durchgeführt werden, damit der Datensatz für das Training geeignet ist. Dies schafft die Überleitung zur nächsten Phase des CRISP-DM-Zyklus, der Data Preparation.

3. Data Preparation

   Wie bereits beschrieben müssen erschlossene Datensätze nach der Akquise an den jeweiligen Anwendungsfall angepasst werden. Die Phase der Datenvorbereitung im CRISP-DM Zyklus widmet sich genau diesem Anliegen. Diese Phase widmet sich der Datenvorbereitung. Im Falle des CRISP-DM stellt dies die finale Konstruktion der finalen Datensätze bzw. des finalen Datensatzes für jeweilige Modellierung dar. Für das Projekt bedeutet dieser Schritt jedoch etwas anderes. Wie bereits angedeutet, müssen Daten zunächst aufbereitet werden, bevor sie Verwendung finden können. Bei einer binären Klassifikation von bspw. Hund oder Katze müssen dem Netzwerk Daten übergeben werden, die mit einer entsprechenden Markierung versehen sind, damit der Algorithmus weiß, ob beim jeweilig übergebenen Bildobjekt eine Katze oder ein Hund abgebildet ist. Diese Form der Markierung werden auch Label genannt. Bevor dem Algorithmus in irgendeiner Form Daten übergeben werden können, müssen die vorhanden Datensätze zuvor gelabelt werden, damit sie für den Algorithmus eindeutig zuzuordnen sind. Zwar wird mit diesem Punkt etwas von der nachfolgenden Phase vorhergegriffen, allerdings trifft dieser eher auf die Data Preparation als auf das Modeling zu. Netzwerkarchitekturen verlangen, dass übergebene Datenobjekte, in diesem Falle Bilder, in einem bestimmten Format vorliegen. Soll heißen, dass bspw. nur Bilder der Größe 224x224 vom Netzwerk verarbeitet werden können und diese dementsprechend zuvor auf die passende Größe zurechtgeschnitten werden müssen. Für das Projekt bedeutet dies, dass entsprechende Methoden bereitgestellt werden müssen, damit die zu verwendenden Datensätze stets in korrekter Form vorliegen. In der späteren Beschreibung jeweiliger Anforderungen wird auf dieses Verfahren ebenfalls näher eingegangen.

4. Modelling

   Allgemein beschreibt diese Phase die Identifizierung und Anwendung geeigneter Data-Mining-Verfahren. Im Falle des Projekts beschreibt diese Phase allerdings im Allgemeinen die Identifizierung, Konstruktion und letztendliche Anwendung einer ge-

eigneten Netzwerkarchitektur zur Lösung entsprechender Klassifizierungsprobleme. Wie bereits angedeutet stellen CNN's die State-of-the-Art-Lösung bei der Verarbeitung von Bildobjekten dar. Allerdings existieren in der Literatur, als auch in der Praxis, viele unterschiedliche Netzwerkarchitekturen, die auf der Grundidee des Konzeptes von CNN's basieren, allerdings unterschiedliche Merkmale aufweisen und in spezifischen Anwendungsfällen bessere Performance aufweisen als andere. Es bestand zudem auch die Möglichkeit, eigene Architekturen zu gestalten und mit verschiedenen Modellen zu experimentieren, allerdings ist es wesentlich unkomplizierter, auf bereits vorhandene Lösungen zurückzugreifen. Innerhalb eines Netzes können des Weiteren unterschiedliche Parameter individuell angepasst werden. Es gilt, passende Hyperparameter zu finden, unterschiedliche Optimizer auszuwählen und passende Loss-Funktionen zu setzen. Über jeweilige Optimizer sowie Loss-Funktionen kann ebenfalls im Anhang nähere Informationen eingeholt werden. Bei der Beschreibung der Anforderungen wird ebenfalls genauer auf zuvor genannte Parameter eingegangen.

5. Evaluation

   Diese Phase überscheidet sich größtenteils mit dem, was auch im Rahmen des Projektes in entsprechender Phase passiert. Hier werden zuvor angewendete Modelle nach ihrer Performance bewertet und entschieden, welches sich für die vorliegende Aufgabenstellung am besten eignet. Im Rahmen des Projektes kam es häufiger vor, dass Netzwerkarchitekturen und -Modelle ausgetauscht wurden, da sich herausstellte, dass zweiteres besser für den jeweiligen Anwendungsfall geeignet ist. Zudem wird innerhalb dieser Phase die Performance des Algorithmus bewertet anhand von zuvor definierten Metriken. An dieser Stelle entscheidet sich, ob ein Netzwerk für die jeweilige Aufgabe geeignet ist, ob potentiell neue Möglichkeiten erschlossen werden müssen oder ob die Datengrundlage ungeeignet ist.

6. Deployment

   Innerhalb dieser Phase werden die jeweilig trainierten und evaluierten Netzwerke deployed, d. h. in den jeweiligen Dienst in der Service-Architektur eingebunden.

9.3 Machine-Learning-Werkzeugkomponenten

Machine Learning bzw. Deep Learning ist ein äußerst komplexes und facettenreiches Themengebiet. Bei der Konstruktion, Planung und Durchführung von Machine Learning Projekten gibt es viele unterschiedliche Aspekte, die beachtet werden müssen. Von der Wahl

der Netzwerkarchitektur, zur Konstruktion eigener spezifischer Modelle mit speziellen Layern bis hin zur Wahl geeigneter Hyperparameter und Optimizer gibt es viele Komponenten, die essenziell zur Durchführung letztendlich erfolgreicher Machine Learning Projekte sind. In diesem Kapitel sollen kurz die jeweiligen Komponenten beschrieben und erklärt werden, welche zur Durchführung der anfallenden Arbeitspakete verwendet wurden. Gleichzeitig soll hiermit eine grundlegende Wissensbasis vermittelt werden, um das im späteren Verlauf beschriebene Vorgehen der jeweiligen Arbeitspakete besser nachvollziehen zu können. Im Rahmen dieser Dokumentation dienen diese Komponenten auch gleichzeitig als eine Art Werkzeugkasten, auf dem im späteren Verlauf innerhalb der jeweiligen Vorgänge referenziert werden kann.

Zunächst jedoch ein paar allgemeine Kontextinformationen. Zwar werden an dieser Stelle einige Inhalte vorgezogen, dies ist jedoch notwendig, um die bereitgestellten Informationen in den folgenden Zeilen besser zuordnen zu können. Betrachtet man einen Machine Learning Anwendungsfall, wird zunächst unterschieden um welche Art von Anwendungsfall es sich handelt. Dabei wird differenziert zwischen einem überwachten Use-Case und einem unüberwachten Use-Case. In den Anwendungsfällen innerhalb der Projektgruppe handelt es sich ausschließlich um überwachte Machine Learning Anwendungsfälle, da der Kern der Zielanforderungen aus Klassifizierungsaufgaben besteht. Supervised bedeutet im Allgemeinen, dass auf der einen Seite ein definierter Input X dem neuronalen Netzwerk zugeführt wird und auf der anderen Seite ein gewünschter Output Y definiert wird, der als letztendlicher Rückgabewert des Netzwerks erwartet wird. Ziel ist es somit, eine Zuordnungsfunktion zu approximieren, die bei Übergabe von Daten des Typs X erfolgreich die gewünschte Zielvariable Y prognostiziert [Bro16]. Im Gegensatz zum überwachten Lernen hat das unüberwachte Lernen nur einen definierten Input X und keinen gewünschten Output Y. In diesem Falle soll der Algorithmus von selbst Korrelationen innerhalb der übergeben Daten identifizieren, es existiert somit kein richtiger und kein falscher Output [Cho15]. Mit dem Hintergrundwissen kann mit der Beschreibung essenzieller Bausteine zur Bearbeitung von Machine Learning Anwendungsfällen begonnen werden.

Grundbaustein für alle anfallenden Machine Learning Projekte innerhalb der Projektgruppe stellt die auf Python basierende Deep Learning Bibliothek Keras dar. Keras ist eine high-level neural network API, welche in der Open Source Bibliothek TensorFlow eingebettet ist. Keras bietet einen einfachen, nutzerfreundlichen und relativ simplen Einstieg in das Themengebiet Deep Learning und wurde auf Empfehlung des Praxispartners als Standardbibliothek für die entsprechenden Machine Learning Projekte innerhalb der

Projektgruppe etabliert. Neben der Nutzerfreundlichkeit bietet Keras eine große Anzahl vorimplementierter Methoden und Funktionen. Über Funktionen und Methodiken zum Preprocessing, einen einfachen Zugriff auf Transfer Learning durch Bereitstellung vortrainierter und vordefinierter Netzwerkarchitekturen über eine Vielzahl vorimplementierter Schichten bietet Keras eine Bibliothek, die für die jeweiligen Anwendungsfälle innerhalb des Projektes mehr als ausreichend sind. Neben den bereits genannten Funktionen und Services bietet Keras noch eine Vielzahl weiterer Komponenten, auf welche an dieser Stelle jedoch nicht weiter eingegangen wird. Für weitere Information wird auf die offizielle Keras Homepage[1] referenziert.

Im vorherig Beschriebenen wurde bereits der Begriff Netzwerkarchitektur erwähnt. Die Wahl oder Konstruktion der Netzwerkarchitektur ist ein essenzieller Baustein zur Bearbeitung eines Machine Learning Projektes. Je nach betrachtetem Anwendungsfall kann die Wahl der jeweiligen Komponenten stark variieren. Grundsätzlich muss nach der Betrachtung des vorliegenden Anwendungsfalles zunächst die Entscheidung getroffen werden, ob ein vordefiniertes Netzwerk verwendet werden soll oder ob eine, womöglich an den Anwendungsfall angepasste, eigene Netzwerkarchitektur definiert werden soll. Auf letztere Option soll erst im späteren Verlauf dieses Kapitels eingegangen werden. Zunächst wird der Fall bei der Wahl eines vordefinierten Netzwerkes betrachtet, was im Allgemeinen auch unter dem Begriff „Transfer Learning“ zusammengefasst wird.

Für Studenten ist das Konzept des Transfer Learning im übertragenen Sinne allgegenwärtig. In Kurs A lernt man X für die entsprechende Abschlussprüfung, in Kurs B können Teile des gelernten aus X angewendet werden. Transfer Learning im Kontext Deep Learning ist deckungsgleich mit dem genanntem Konzept. Im Kern geht es darum, Wissen aus abweichenden Domänen auf eine andere Domäne zu übertragen. Eines der bekanntesten Beispiele für Transfer Learning stellt ImageNet dar. Kurz ein paar allgemeine Informationen zu ImageNet: ImageNet ist ein Open Source Projekt, welches unter anderem eine große Menge an Bilddateien für wissenschaftliche Forschungszwecke bereitstellt [KSH12].

[Rus+15a] stellte einen Meilenstein für Deep Learning im Bereich Computer Vision dar, wo mithilfe von Deep Learning die ImageNet Large Scale Visual Recognition Challenge vom Team hinter dem zuvor genannten Paper mit großem Vorsprung gewonnen wurde. Wenn sich also im Rahmen dieser Dokumentation auf Transfer Learning bezogen wird, wird damit auch indirekt auf ImageNet referenziert. Zurück zu Transfer Learning. Transfer Lear-

[1] Website von Keras. URL: https://keras.io/ Abrufdatum: 23.04.2019

ning stellt die Disziplin dar, ein vortrainiertes Netzwerk mit entsprechenden vortrainierten Gewichten zu verwenden, um diese auf einen differenzierten Anwendungsfall zu übertragen. Es ist deshalb vortrainiert, da es zuvor auf einem meist sehr großem Datensatz mit sehr vielen Iterationen trainiert wurde und diese Gewichte abspeichert. Betrachtet man nun den eigenen, vorliegenden Anwendungsfall, können die Informationen und Features, die in den Gewichten des jeweiligen vortrainierten Netzwerk abgelegt sind, dazu genutzt werden, um die allgemeine Performance des eigenen Netzwerkes zu steigern. Performance bezieht sich in diesem Falle nicht nur auf die letztendliche Performance des Netzes bezogen auf die Genauigkeit der Prognosen, sondern auch auf die Trainingsgeschwindigkeit oder die Anzahl benötigter Iterationen bis der Algorithmus konvergiert. Obwohl das jeweilig gewählte vortrainierte Netzwerk zwar auf Basis unterschiedlicher und domänenfremder Daten trainiert wurde, können dennoch potenziell zuvor identifizierte und extrahierte Features auf den eigenen Anwendungsfall übertragen werden. Somit kann selbst mit einer limitierten Datenbasis robuste Ergebnisse erzielt werden und stellen einen potenten Startpunkt für den eigenen Anwendungsfall dar [Gup17b].

Bevor die jeweils verwendeten, vordefinierten Netzwerkarchitekturen aus der Keras Bibliothek beschrieben werden, soll zunächst auf die verwendeten Custom Netzwerkarchitekturen eingegangen werden. Vor allem zu Beginn des Projekts und bei der Bearbeitung der ersten Machine Learning Projekte wurde zunächst auf eigens zusammengestellte Custom Modelle zurückgegriffen. Mithilfe der von Keras bereitgestellten Funktion des Sequential Models [AC19] wurden verschieden vordefinierte Layers der Keras Bibliothek aneinandergehangen, um eine Netzwerkarchitektur zu definieren. Wo vordefinierte Netzwerkarchitekturen wie bspw. InceptionNet aus teils zwei Dutzend Schichten bestehen, enthalten die im Rahmen des Projektes, eigens definierten Netze eine variierende Anzahl von Schichten, welche aber in keinem der Fälle den zweistelligen Bereich überschreitet. Die Keras Bibliothek stellt dabei viele vordefinierte Schichten zum Importieren bereit, welche lediglich in das jeweilige Projekt integriert werden müssen und anschließend mithilfe der Sequential Model-Funktion an die jeweilige Architektur gekettet werden können. Keras stellt eine Vielzahl vordefinierte Layer zur Verfügung. Nicht alle der Layer-Optionen finden jedoch Anwendung innerhalb des Projekts, viele der Optionen sind anwendungsspezifisch und eigenen sich dementsprechend nur für jeweilige Domänen. Im Kontext des Projektes sind vor allem Komponenten aus dem Bereich Computer Vision relevant. Aus diesem Grund finden neben den bereitgestellten Core-Schichten [CA19c] vor allem Convolutional Layers [CA19b] sowie Pooling Layers [CA19d] aus der Keras Bibliothek Anwendung innerhalb des Projekts.

Im Folgenden sollen, je nach Kategorie, entsprechende Schichten aufgelistet werden, welche innerhalb des Projektes Verwendung finden. Anschließend soll zudem kurz beschrieben werden, welche Funktionen diese erfüllen und warum sie Anwendung finden.

Core-Layers

- Dense
  Eine Dense-Layer ist gleichzusetzten mit einer einfachen Fully Connected Layer in einem Neuronalen Netzwerk. Dies bedeutet lediglich, dass jedes Neuron der aktuell betrachteten Schicht einen Input von allen Neuronen der vorherigen Schicht erhält, dementsprechend vollständig vernetzt ist. Die Dense-Layer ist abseits der vollständigen Vernetzung eine Schicht ohne spezielle Funktionen oder Merkmale. Benötigte, bzw. optionale Argumente, die bei der Deklarierung überreicht werden können, können in der bereitgestellten Dokumentation von Keras, siehe [CA19c], tiefergehend betrachtet werden. Dies gilt auch für alle zukünftig aufgeführten Schichten.

- Dropout
  Die Verwendung von Dropout Schichten hat das Ziel, Overfitting zu vermeiden bzw. einzugrenzen. Overfitting im Kern bedeutet, dass das vorliegende Model zu gut an die Trainingsdaten gewöhnt ist, jedoch Probleme beim Generalisieren auf bspw. einem vorliegenden Validierungsset hat. Eine Dropout Layer erfüllt die Funktion, eine zuvor definierte Prozentzahl an zufälliger Input Units auf null zu setzten. Im übertragenen Sinne wird somit ein gewisser Prozentsatz zufälliger Neuronen im Netzwerk deaktiviert um für zusätzliche Dynamiken bei den Verrechnungen innerhalb des Netzwerkes zu sorgen. Weitere Informationen zum Thema Overfitting können in [Sri+14] eingesehen werden.

Pooling Layers

- MaxPooling
  MaxPooling ist ebenfalls eine Methode, Overfitting im betrachteten Netzwerk zu vermeiden. Neben der Vermeidung von Overfitting reduzieren MaxPooling Layers allerdings auch die Berechnungskosten, die pro Interation aufgewendet werden müssen, indem die Anzahl der zu berechnenden Parameter reduziert wird. Um dies zu erreichen, reduzieren MaxPooling Layers die Dimensionen eines übergebenen Inputs

wie bspw. Bildobjekte. Somit kann neben der Vermeidng von Overfitting zudem eine Performancesteigerung bezogen auf die Trainingsdauer pro Iteration erzielt werden [CA19d]. Eine detaillierte Beschreibung zu MaxPooling Layern findet sich in der Seminararbeit zu Convolutional Neural Networks, welche im Anhang zu finden ist.

Convolutional Layer

- Convolutional Layer finden vor allem Einsatz in Anwendungsfällen im Bereich Computer Vision, also der Verarbeitung statischer Bildobjekte. Da es sich in den Teilprojekten innerhalb des Brillenberaters vor allem um Klassifizierungen auf Basis von Bilddateien handelt, ist es selbsterklärend warum diese Art von Layer [CA19b] Anwendung in den jeweiligen Costum Modellen findet. Eine detaillierte Erklärung wie Convolutional Layer bzw. Convolutional Neural Networks funktionieren, kann ebenfalls in der Seminararbeit zu Convolutional Neural Networks im Anhang eingesehen werden.

Damit sind alle innerhalb der Custom Modelle verwendeten Schichten abgedeckt und kurz erläutert. Nun soll die Auflistung und kurze Erläuterung der von Keras bereitgestellten, vordefinierten Netzwerkarchitekturen folgen. Wie bereits erwähnt dienen diese bereitgestellten Netzwerkarchitekturen der Möglichkeit, Transfer Learning anzuwenden. Keras bietet unterschiedliche Architekturen, die in das jeweilige Projekt definiert und eingebettet werden können. Im Folgenden ist die Abbildung 6 zu sehen, die eine Sammlung aller verfügbaren vordefinierten Netzwerkarchitekturen aufzeigt. Neben den Modellnamen, der Größe und der Anzahl der Parameter wird zudem auch die Top-1- sowie Top-5-Accuracy des jeweiligen Netzwerks aufgezeigt.

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
ResNeXt50	96 MB	0.777	0.938	25,097,128	-
ResNeXt101	170 MB	0.787	0.943	44,315,560	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

Abbildung 6: Sammlung aller verfügbaren vordefinierten Netzwerkarchitekturen [CA19a]

Netzarchitekturen, welche Anwendung innerhalb des Projektes verwendet werden, werden im Folgenden kurz aufgelistet.

- VGG16
- ResNet50
- InceptionV3
- DenseNet
- MobileNet

Ohne tiefergehend auf die allgemeine Struktur und den Aufbau der jeweiligen Architekturen einzugehen (dazu können in den jeweils referenzierten Quellen nähere Informationen eingeholt werden), soll kurz erläutert werden, warum sich für jeweilige Netzwerkarchitekturen entschieden wurde. Neben der allgemeinen Experimentierfreude, war auch die Größe und die Performance des jeweiligen Netzwerks ausschlaggebend für die letztendliche Wahl. Wie bereits erwähnt kann in obiger Abbildung 6 die Top-1 sowie Top-5 Accuracy eingesehen werden. Diese Metriken werden häufig dazu verwendet, die Performance von

Netzwerken bei Klassifizierungsaufgaben zu evaluieren. Der Top-1 Score beschreibt dabei den Fall, in dem das Netzwerk das jeweilige Klassifizierungsproblem korrekt prognostiziert, bzw. bei der Ausgabe mehrerer Ergebnisse das Ergebnis mit dem höchsten Wahrscheinlichkeitswert, welches mit dem tatsächlich erwarteten Wert übereinstimmt. Die Top 5-Accuracy wiederum sagt aus, ob die jeweilig korrekte Prognose unter den fünf besten Prognosen des Netzwerkes liegt, bzw. unter den fünf Prognosen mit dem jeweils höchstem Wahrscheinlichkeitswert. Neben der generellen Performance der jeweiligen Netzwerkarchitektur war auch die jeweilige Größe des Netzes von Relevanz. Im Laufe des Projektes hat sich herausgestellt, dass die im Projekt verwendete Architektur im Backend Probleme bei der Verarbeitung von Netzwerken hat, die mehrere hundert MB groß sind. Deshalb mussten Anpassungen in einigen Arbeitspaketen vorgenommen werden, doch dazu mehr im späteren Verlauf.

Die Konstruktion benutzerdefinierter Architekturen, die dazu verwendeten Layer und auch die Implementierung vordefinierter Netzwerkarchitekturen wurde damit beschrieben. Im Rahmen eines Machine Learning Projektes ist damit allerdings zunächst nur das Fundament für das weitere Vorhaben gelegt. Wie bereits beschrieben ist die Wahl geeigneter Hyper- als auch Standardparameter, Aktivierungsfunktionen, Loss-Funktionen sowie die Selektion passender Optimizer ein weiterer essenzieller Schritt bei der Bewältigung derartiger Projekte. Genannte Komponenten stellen in diesem Fall die Nuancen und Feinheiten dar, um auf Grundlage des gelegten Fundaments funktionierende Lösungen für den betrachteten Anwendungsfall zu schaffen. Im Folgenden sollen folgende Komponenten kurz beschrieben werden und anschließend eine Auflistung der Komponenten folgen, welche im Projekt Anwendung gefunden haben.

Aktivierungsfunktion

Zuvor wurde bereits erwähnt, dass jedes Neuron innerhalb eines definierten Netzwerkes sogenannte Gewichte berechnen und letztendlich in den jeweiligen Zellen abspeichern. Bei der Beschreibung der Fully-Connected-Layer wurde dieses Prinzip verdeutlicht. Hier erhält das Neuron der aktuell betrachteten Schicht einen Input von allen Neuronen der vorherigen Schicht. Es folgt eine simple Berechnung des Inputs des bzw. der Neuronen der vorherigen Schicht multipliziert mit dem Gewicht des aktuell betrachteten Neurons. Diese Gleichung entspricht der Gleichung einer linearen Regression. Ein neuronales Netzwerk mit einem einzigen Neuron wäre in diesem Falle nichts anderes als die Berechnung einer einfachen linearen Regression. Den entscheidenden Unterschied stellt jedoch dabei die anschließende

Verrechnung mithilfe der gewählten Aktivierungsfunktion dar. Bevor die berechneten Gewichte an die Neuronen der nächsten Schicht weitergebeben können, werden diese durch eine Aktivierungsfunktion modifiziert. Die Aktivierungsfunktion bestimmt, ob ein jeweiliges Neuron einen Schwellenwert überschreitet und letztendlich aktiviert wird oder nicht. Im übertragenem Sinne wird den Neuronen damit mitgeteilt, ob eine jeweilige Information oder ein jeweilig identifiziertes Feature für den Gesamtkontext von Relevanz ist und weiter berücksichtigt werden sollte oder ob die Information ignoriert werden kann. Aktivierungsfunktionen zeichnen sich dadurch aus, dass sie, anders als die vorherige lineare Berechnung der Gewichte, nicht linear sind und somit wesentlich komplexere Zusammenhänge abbilden können [Gup17a].

Die Keras Bibliothek stellt eine Vielzahl vorimplementierter Aktivierungsfunktionen bereit, die lediglich in das jeweilige Projekt integriert werden müssen. Im Projekt fanden unterschiedliche Aktivierungsfunktionen Anwendung. Die wohl bekannteste Aktivierungsfunktion stellt die Sigmoid-Funktion dar. Für die Anwendungsfälle innerhalb des Projektes stellte sich jedoch die Aktivierungsfunktion ReLU als beste Alternative dar. Im Folgendem soll eine kurze Auflistung und anschließende kurze Beschreibung der im Projekt verwendeten Aktivierungsfunktionen erfolgen.

- Sigmoid

Die Sigmoid-Funktion erhält einen Wert als Input und modifiziert ihn in einen Wert, der im Bereich zwischen 0 und 1 liegt. Kerneigenschaft der Sigmoid Funktion ist, dass sie kontinuierlich bzw. an jeder Stelle differenzierbar ist. Ein weiterer Vorteil ist, dass die Werte der jeweiligen verrechneten Aktivierungen stets in einem Bereich zwischen 0 und 1 liegen, sodass das Risiko explodierender Werte vermieden wird. Ein zentrales Problem der Sigmoid-Funktion ist das sogenannte Vanishing Gradient Problem [FP18a; Gup17a]. Mehr dazu in der Seminararbeit zu Recurrent Neural Networks im Anhang.

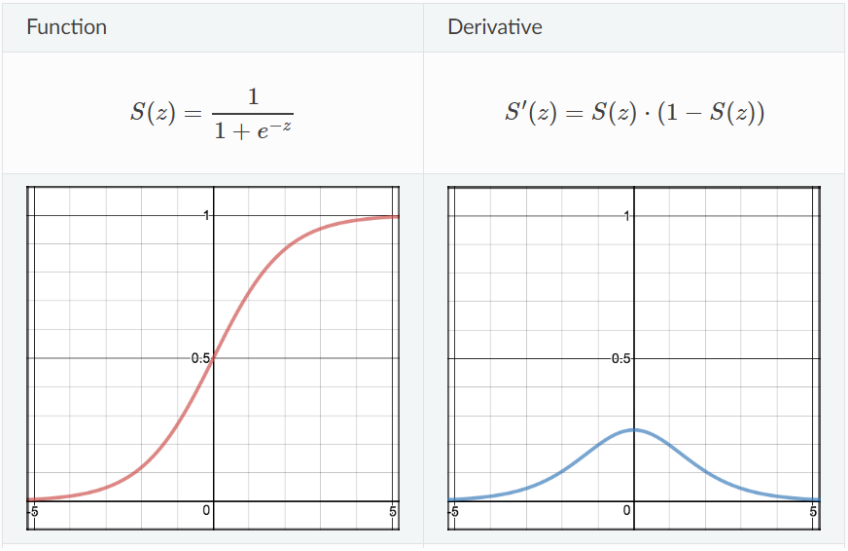


Abbildung 7: Sigmoid-Funktion [FP18a]

- Tanh

Anders als die Sigmoid-Funktion, welche einen Wertebereich von [0;1] abdeckt, deckt die Tanh-Funktion einen Wertebereich von [-1;1] ab. Insgesamt ist sie der Sigmoid-Funktion sehr ähnlich, im übertragenen Sinne stellt die Tanh nur eine skalierte Version der Sigmoid-Funktion dar. Zentraler Vorteil gegenüber der Sigmoid-Funktion ist, dass negative Werte direkt in negativen Bereichen gemappt werden können und Werte nahe Null werden entsprechend zugeordnet. Zwar ist das Problem des Vanishing Gradient im Kontext der Verwendung der Tanh-Funktion nicht so ausgeprägt, allerdings ist es immer noch präsent [FP18b; Gup17a].

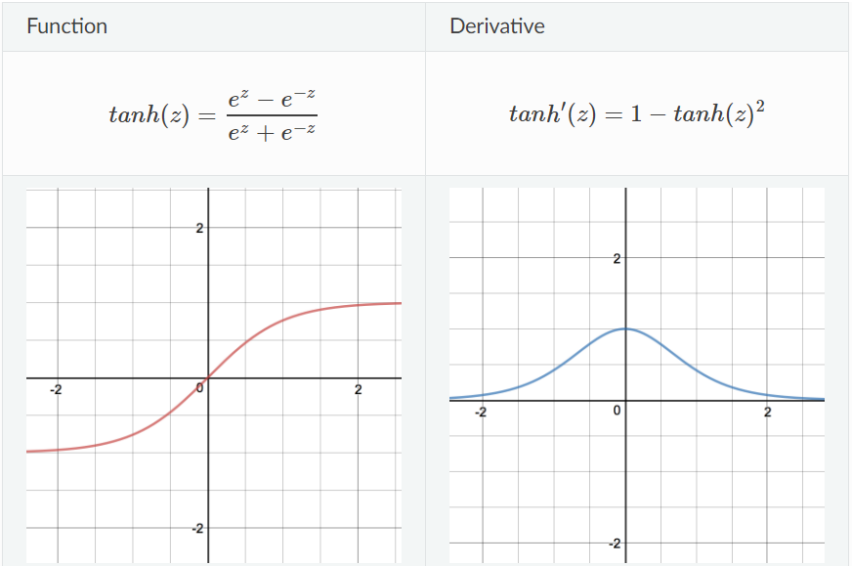


Abbildung 8: Tanh-Funktion [FP18b]

- ReLU

ReLU, bzw. Rectified Linear Unit, stellt die zur Zeit am häufigsten genutzte Aktivierungsfunktion dar. Vor allem bei der Verwendung innerhalb Convolutional Neural Networks haben sie sich als neuen Standard etabliert. Im Kern bieten ReLU-Funktionen die selben Vorteile einer Sigmoid-Funktion, allerdings mit wesentlich besserer Performance [FP18c; Gup17a]. Detaillierte Informationen zu ReLU können in der Seminararbeit zu Convolutional Neural Networks eingeholt werden.

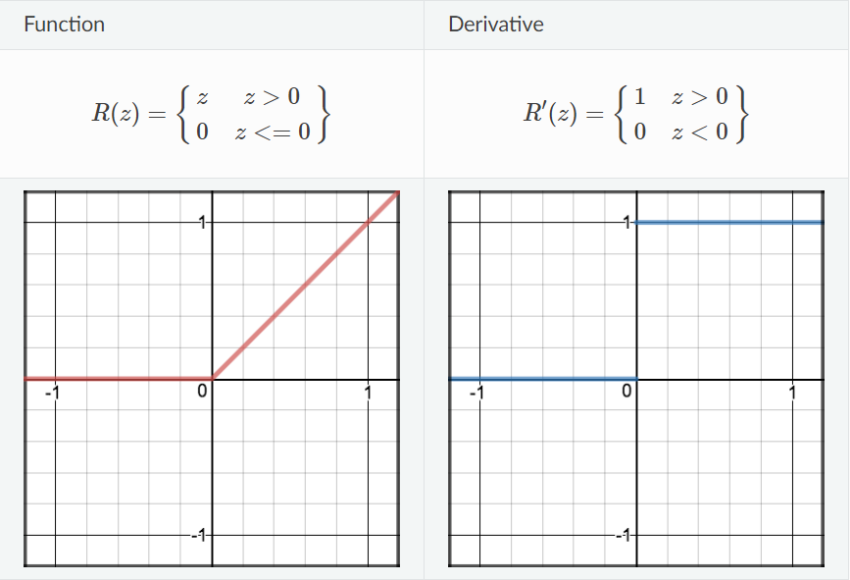


Abbildung 9: ReLU-Funktion [FP18c]

- Softmax

Die Softmax-Funktion berechnet die Wahrscheinlichkeitsverteilung des Ergebnisses über „n“-Ergebnisse. Im Allgemeinen berechnet die Funktion die Wahrscheinlichkeit jeder Zielklasse über alle möglichen Zielklassen. Diese berechneten Wahrscheinlichkeiten dienen dazu, die jeweiligen Prognosen des Algorithmus zu bewerten und eine finale Zuordnung abzuleiten. Es findet vor allem Anwendung bei der Klassifizierung mehrerer Objektklassen [Gup17a; FP18d].

Nachdem das Konzept der Aktivierungsfunktion, deren genereller Nutzen und die im Projekt verwendeten Aktivierungsfunktionen aufgeführt wurden, folgt nun das Konzept der Loss-Funktion. Wie bereits beschrieben wurde, wird nach der Berechnung der Gewichte innerhalb eines Neurons die Aktivierungsfunktion angewendet, um die Aktivitätslevel des jeweiligen Neurons zu bestimmen. Dieser Prozess zieht sich durch die gesamte Struktur des neuronalen Netztes, bis zur Verrechnung in der letzten Schicht des Netzes. Dort werden zusätzlich Verrechnungen angewendet, um die letztendliche Performance des Netzes in der aktuell betrachteten Iteration zu bewerten. Diese Berechnung erfolgt durch die Loss-Funktion. Eine Loss-Funktion berechnet einen Realwert, aus welchem abgeleitet werden kann wie gut das Netzwerk bzw. der Algorithmus in der aktuellen Iteration auf das jeweilige Problem angepasst ist. Fällt dieser Wert eher hoch aus, so bedeutet das, dass

der Algorithmus noch nicht gut genug an den jeweiligen Anwendungsfall angepasst ist [FVK18a]. Das Ziel ist es, diesen Wert weitestgehend einem Minimum anzunähern, sodass der Algorithmus bestmöglich auf die jeweilige Problemstellung abgeglichen ist. Deshalb werden an dieser Stelle eingehend mit der Loss-Funktion auch die sogenannten Optimizer beschrieben.

Optimizer dienen dazu, den Fehlerwert der Loss-Funktion zu reduzieren gegen Minimum. Im Kern geht es bei einem überwachten Machine Learning Anwendungsfall darum, den Fehlerwert der Prognose bezogen auf den gewünschten Output zu minimieren. Der Output der Loss-Funktion der jeweilig betrachteten Iteration beschriebt einen zusammengefassten Fehlerwert, den der Algorithmus beim vorliegendem Klassifizierungsproblem aufweist. Der Optimizer soll nun dafür sorgen, dass dieser Fehlerwert am Ende der nächsten Iteration reduziert wird. In Abhängigkeit des zuvor berechneten Loss werden die in der Iteration ermittelten Gewichte durch den Optimizer dahingehend angepasst, dass der letztendliche Fehlerwert der folgenden Iteration geringer ausfällt als zuvor. Es folgt dementsprechend eine Aktualisierung bzw. Optimierung der im Netzwerk vorliegenden Parameter, um den Loss zukünftiger Iterationen zu minimieren. Dies ist zudem der Grund, warum Loss-Funktion sowie Optimizer zusammen betrachtet werden. Die Synergie bzw. die Zusammenwirkung beider Komponenten sind letztendlich entscheidend dafür, die Performance der Prognosen zu verbessern und das „Lernen“ innerhalb eines neuronalen Netzes zu ermöglichen. Der berechnete Fehlerwert der Loss-Funktion vermittelt dem Optimizer eine Art Leitfaden, an den er sich halten kann, um die aktualisierten Parameter im Anschluss der Iteration zu bewerten. Wird der Fehlerwert reduziert, bedeutet dies im Umkehrschluss, dass der Algorithmus im jeweilig betrachtetem Anwendungsfall besser performt [Alg18].

Die verwendete Keras Bibliothek bietet eine Vielzahl an vorimplementierten Loss-Funktionen und Optimizern. Wie schon bei den zuvor genannten Aktivierungsfunktionen finden im Falle der Loss- bzw.- Optimizer-Funktionen ebenfalls nicht alle Optionen im Projekt Anwendung. Im Folgenden soll deshalb eine Auflistung und anschließende kurze Beschreibung aller verwendeten Loss-Funktionen sowie Optimizer erfolgen.

Loss-Funktionen

- Binary Cross Entropy

Diese Art von Loss-Funktion findet vor Allem Anwendung in Projekten, in denen der Anwendungsfall durch ein binäres Klassifizierungsproblem beschrieben wird. Einfaches Bei-

spiel dafür wäre die Unterscheidung zwischen Hund und Katze, oder dem projektbezogenen Fall der Unterscheidung zwischen Mann und Frau. Wie bereits beschrieben handelt es sich bei einem überwachtem Machine Learning Anwendungsfall um die Zuordnung eines Eingabeobjekts X auf einen Output Y. In einem binären Klassifizierungsproblem handelt es sich im Kern um eine Zuordnung des Inputs auf jeweils zwei Zielklassen. Cross-Entropy gibt als Output einen Realwert zwischen 0 und 1 zurück, der anschließend als Wahrscheinlichkeitswert zur Zuordnung des betrachteten Objekts auf der Zielklasse interpretiert werden kann. Die jeweiligen Extrempunkte [0;1] stellen die beiden Zielklassen dar, der Output der Cross-Entropy-Funktion dient nun dazu, die letztendliche Prognose des Netzwerkes auf einen der beiden Extremwerte zuzuordnen [FVK18b].

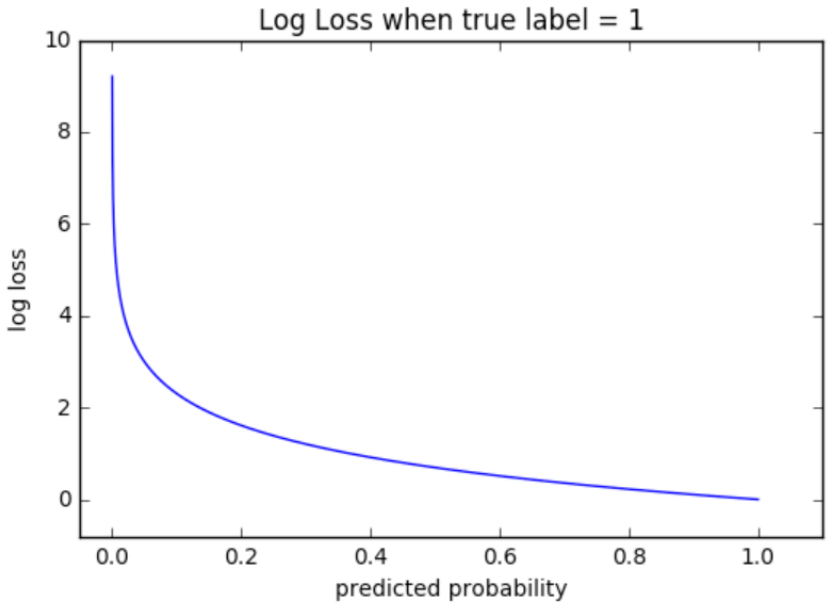


Abbildung 10: Binary Cross Entropy [FVK18b]

- Categorical Cross Entropy

Binary Cross Entropy dient dazu, in Anwendungsfällen binärer Klassifizierungsprobleme den Output mithilfe eines Realwertes auf eine der jeweiligen Zielklassen zu mappen. Liegt allerdings ein Multi-Klassifizierungsproblem vor, also der Zuordnung eines Inputs auf mehrere, disjunkte Zielklassen, muss auf die Categorical Cross-Entropy zurückgegriffen werden. Diese erfüllt genau die Anforderungen, um zuvor genannte Anforderung bei der Zuordnung auf mehrere Zielklassen zu ermöglichen.

- Mean Squared Error

- Mean Absolute Error
- LogCosh

Optimizer

- SGD
- ADAM

Datasets

Wie bereits im allgemeinen Vorgehen zu Machine Learning Projekten beschrieben, stellen Daten die zentrale Grundlage eines jeden Machine Learning Anwendungsfalles dar. Es müssen passende, auf den Anwendungsfall zugeschnittene Datenquellen identifiziert werden und entsprechendes Domänenwissen aufgebaut werden. Innerhalb der Teilprojekte des Brillenberaters wurden verschiedene Datenquellen gesichtet. Jeder dieser Datensätze weist unterschiedliche Eigenschaften auf und grenzt sich zudem durch die jeweilige Größe, Aufbau, Struktur, aber auch durch deren Qualität voneinander ab. Im Folgenden soll eine Auflistung jener Datensets erfolgen, welche Anwendung innerhalb des Projektes gefunden haben. Zusätzlich zu einer kurzen Beschreibung der wichtigsten Fakten zu jeweiligem Datensatz, soll zudem auch die entsprechende Referenz angegeben werden, um bei Interesse spezifischere Informationen zur jeweiligen Datenquelle einholen zu können.

UTKFace [ZSQ17]:

- Datenset mit Gesichtern unterschiedlicher Personen und unterschiedlicher Altersgruppen
- Alter der Personen reicht von 0 bis 116
- Annotationen für Alter, Geschlecht, Ethnizität und Landmarks
- Besteht aus ca. 20.000 Bildern, jedes der Bilder enthält nur ein Gesicht
- Bilder bereits korrekt ausgerichtet und zurechtgeschnitten (was das Preprocessing vereinfacht)
- Link: https://susanqq.github.io/UTKFace/

Labeled Faces in the Wild (LFW) [Lea+16]:

- Datenset bestehend aus ca. 13.000 Bildern
- Ca. 6.000 unterschiedliche Personen, ca. 1.700 Personen mit zwei oder mehr Bildern
- Annotationen für eine Vielzahl unterschiedlicher Features, wie Geschlecht, Haarfarbe, Ethnizität, ob eine Sonnenbrille getragen wird, ob eine normale Brille getragen wird, ob die Augen offen oder geschlossen sind etc. (mehr Informationen über zusätzliche Annotationen finden sich auf der Homepage)
- Bietet verschiedene Versionen des Datensets (funneled, deep-funneled, wurde allerdings im Projekt nicht verwendet)
- Link: http://vis-www.cs.umass.edu/lfw/index.html#information

IMDB-Wiki [RTG16]:

- Bilder gecrawlt von Wikipedia und IMDB
- Ca. 500.000 Bilder von Gesichtern mit Alter und Geschlecht als Annotation
- Enthält zudem Metadaten zu jedem Bild wie Name oder Zeitpunkt des Fotos
- Bis dato größtes Datenset für den Anwendungsfall der Altersprognose
- Bieten Datensatz als Rohfassung oder bereits zurechtgeschnitten und ausgerichtet (was den benötigten Speicher deutlich reduziert)
- Link: https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/

APPA-REAL [Agu+17]:

- Datenset bestehend aus ca. 8.000 Bildern
- Annotation für tatsächliches sowie scheinbares Alter
- Ebenfalls Annotationen für Geschlecht und Ethnizität
- Enthält sowohl die „rohe“ Datei als auch bereits zurechtgeschnittene und skalierte Bilddateien
- Annotationen in separater CSV-Datei abgelegt, welche einfach in das jeweilige Projekt eingebunden werden kann

- Link: http://chalearnlap.cvc.uab.es/dataset/26/description/

YouTube-Faces-With-Facial-Keypoints [BT17]:

- Datenset, welches aus ca. 1300 Videos mit variierender Länge besteht, in den jeder Frame des Videos mit Annotationen für die Facial-Landmarks der im Video gezeigten Person enthält
- Annotationen sowohl für 2D- als auch 3D Facial-Landmarks
- Link: https://www.kaggle.com/selfishgene/youtube-faces-with-facial-keypoints

9.4 Machine-Learning-Verfahrenskomponenten

Neben dem Machine-Learning Werkzeugkasten, in dem verschiedene Komponenten beinhaltet sind, welche regelmäßig Anwendung in jeweiligen Anwendungsfällen finden, existieren auch beim allgemeinen Ablauf der Konstruktion eines Anwendungsfalles bestimmte Arbeitsabläufe, die sich in allen bearbeiteten Machine-Learning-Projekten wiederfinden. Von den jeweilig benötigten Imports über Methoden zum Zurechtschneiden und Zentrieren übergebener Bilder bis hin zu Hilfsmethoden zum batchweisen Laden der Datensets gibt es Ansätze, die (zwar in stets auf den Anwendungsfall adaptierter Form) in jedem der bearbeiteten Machine-Learning-Projekte Anwendung finden. Dieses Kapitel soll dazu dienen, eine Übersicht dieser Techniken und Methoden zu liefern, auf welche sich in der Beschreibung des Vorgehens des jeweiligen Projektes bezogen werden kann, um doppelt formulierte Ausführungen zu vermeiden. Dabei sollen beispielhafte Aufnahmen aus jeweiligen Jupyter Notebooks bei der Beschreibung des Aufbaus eines Machine-Learning-Projektes hinzugezogen werden, um zusätzlichen Kontext herzustellen.

Die allgemeine Projektstruktur innerhalb der Jupyter Notebooks gliedert sich wie folgt. Zunächst wird das Preprocessing durchgeführt. Dies umfasst zum einen die Explorationen des zu verwendeten Datensatzes, als auch bspw. die Vorbereitung des Datensatzes auf den anstehenden Trainingsprozess.

```
import numpy as np
import pandas as pd
import matplotlib.image as mpimage
import matplotlib.pyplot as plt
import matplotlib
import glob
import cv2
import pickle
from PIL import Image
```

Abbildung 11: Bibliotheken und Frameworks, die Grundlage für die meisten Preprocessing-Prozesse waren

In der oben gezeigten Abbildung 11 werden die Bibliotheken und Frameworks aufgeführt, welche Grundlage für die meisten Preprocessing-Prozesse waren. Es ist anzumerken, dass Bibliotheken wie NumPy oder cv2 nicht nur in den Preprocessing-Phasen Anwendung gefunden haben, sondern auch für weitere Arbeitsschritte eingesetzt wurden. NumPy[2] stellt dabei ein wesentliches Tool dar, welches die vielzähligen Matrizenmanipulationen so einfach und unkompliziert wie möglich gestaltet. Ohne tiefergehend auf die Funktionsweise sowie Funktionsumfang der Bibliothek einzugehen, ist jedoch hinzuzufügen, dass NumPy eines der wichtigsten Tools zur Bearbeitung von Machine-Learning Projekten darstellt. Näheres kann in [Oli06] eingesehen werden.

Pandas[3] bietet eine Vielzahl an Möglichkeiten, bestehende Daten zu strukturieren, aufzubereiten und zu analysieren. Mithilfe von Pandas konnten Dataframes erstellt werden, welche zum einen die jeweiligen Metainformationen der verwendeten Bilder wie bspw. Namen oder Speicherpfad enthielten und zum anderen die jeweiligen Annotationen des zugehörigen Bildes ebenfalls hinzufügten. Somit konnten mit Dataframes die relevanten Informationen der meist separat verwalteten Daten- sowie Annotationsinformationen mit vergleichsweise geringem Aufwand zusammengeführt werden. Des Weiteren war die Bibliothek sehr hilfreich dabei, jeweilige Datensätze zu explorieren und verschiedene Visualisierungen vorzunehmen. Nähere Informationen sind in [McK10] zu lesen.

Matplotlib[4] ist eine weitere Library, die sowohl zum Laden der Bilddateien genutzt wurde, als auch um diese Dateien in jeweiligen Jupyter Notebooks entsprechend zu visualisieren. Ebenfalls nützlich war die Funktion innerhalb weniger Zeilen Code einfache Graphen plotten zu lassen, um beispielsweise den Trainingsverlauf des Netzes visualisieren zu können und diese anschließend für die Evaluation des Netzes heranziehen zu können. Matplotlib

[2]Dokumentation von NumPy. URL: https://www.numpy.org/devdocs/ Abrufdatum: 23.04.2019
[3]Website von Pandas. URL: https://pandas.pydata.org/ Abrufdatum: 23.04.2019
[4]Website von Mathplotlib. URL: https://matplotlib.org Abrufdatum: 23.04.2019

bietet zudem eine Vielzahl weiterer Funktionen. Für nähere Informationen kann [Hun07] eingesehen werden.

Restliche aufgeführte Imports dienen ebenfalls hauptsächlich zur Bearbeitung und Manipulation der zum Training benötigten Daten. Die Pickle-Bibliothek diente bspw. oft dazu, die in NumPy-Arrays umgewandelten Bilder sowie Metadaten zu serialisieren bzw. zu deserialisieren, um diese speicherschonend laden bzw. speichern zu können.

Nachdem jeweilige Datenvorbereitungsprozesse abgeschlossen waren, konnte mit der Konstruktion eines Netzwerks sowie mit der Vorbereitung des Trainings begonnen werden. Dazu wurden zunächst benötigte Imports in das Projekt vorgenommen, welche beispielhaft in der unten aufgeführten Abbildung 12 betrachtet werden können.

```
import glob, os
import keras_applications
import random
import keras.backend as K
import tensorflow as tf
import random
import itertools

from collections import OrderedDict
from collections import Counter
from keras.callbacks import Callback
from keras.models import Model, Sequential
from keras.models import load_model
from keras.optimizers import Adam, SGD
from keras.metrics import binary_crossentropy
from keras.layers import Activation, Dropout, Flatten, Dense, Conv2D, MaxPooling2D, Activation, Input, GlobalAveragePooling2D
from keras.applications.mobilenetv2 import MobileNetV2
from keras.applications.densenet import DenseNet121
from keras.applications.xception import Xception
from keras.applications.inception_v3 import InceptionV3
from keras.applications.nasnet import NASNetMobile
from keras.preprocessing.image import ImageDataGenerator
from keras.utils.generic_utils import CustomObjectScope
from keras.layers import DepthwiseConv2D
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
```

Abbildung 12: Imports für die Netzwerkerstellung

Hier finden vor allem die nötigen Imports der Komponenten und Bauteile statt, aus der letztendlich ein jeweiliges Netzwerk erstellt werden soll. Oben befinden sich übergreifende Imports aus den benötigten Bibliotheken wie Keras und Tensorflow, sowie Imports von Hilfsbibliotheken wie random und itertools, die die Bearbeitung und Erstellung von Hilfsmethoden vereinfachen. Darauf folgen viele unterschiedliche Imports aus der bereitgestellten Keras-Bibliothek. Callbacks dienen bspw. dazu, den Trainingsfortschritt des Netzwerkes besser nachvollziehen zu können, indem es in Kombination mit Tensorboard Variablen wie den Loss sowie Validation-Loss zusammen in einem Graphen plottet, um nach Trainingsende potentielle Erkenntnisse zu ziehen. Des Weiteren bieten Callbacks Funktionen

wie das Early-Stopping, was den Trainingsprozess des Netzwerkes beim Eintreten von Overfitting automatisch abbricht und somit keine weiteren Iterationen mehr zulässt.

Zudem finden sich hier die Imports der Sequential API zum Erstellen und Bearbeiten von Modellen wieder, Imports für die im Projekt verwendeten Optimizer wie ADAM und SGD sowie der Import für die gewählte Metrik zum Evaluieren der Modellperformance, in diesem Falle die „binary_crossentropy“. Neben den zusätzlichen Imports verschiedener Layer, die mithilfe der Sequential API von Keras zu Netzwerken verknüpft werden können, wie Dropout, Dense oder MaxPooling finden sich hier auch die Imports der vorimplementierten Netzwerkarchitekturen wieder. In diesem Projekt wird bspw. die MobileNet- sowie Inception-Architektur verwendet. Zwar kann am Ende nur eine der jeweiligen Architekturen gewählt werden, jedoch wurde innerhalb der Projekte stets mit den unterschiedlichen Architekturen experimentiert, um die bestmögliche Alternative für den betrachteten Anwendungsfall zu finden.

```
#split dataset
shuffle = list(zip(image_data, landmark_data))
random.shuffle(shuffle)
x, y = zip(*shuffle)
x = np.asarray(x)
y = np.asarray(y)
n_images = len(x)
TRAIN_TEST_SPLIT = 0.8
split_index = int(TRAIN_TEST_SPLIT * n_images)
x_train = x[0:split_index]
y_train = y[0:split_index]
x_test = x[split_index:]
y_test = y[split_index:]

print("Anzahl Trainingsdaten_Images: ", len(x_train))
print("Anzahl Trainingsdaten__Label: ", len(y_train))
print("Anzahl Testdaten_Images: ", len(x_test))
print("Anzahl Testdaten_Label: ", len(y_test))
```

Abbildung 13: Beispiel für die Verwendung der Pyhthon-Bibliothek „random“

Im obig gezeigten Ausschnitt (Abbildung 13) wird die Verwendung der zuvor erwähnten Python-Bibliothek „random“ zum Aufteilen des Datensatzes in jeweils Trainings- sowie Testdatenset aufgezeigt. Das Trainingsset wird anschließend dazu verwendet, das auf den vorliegenden Anwendungsfall angepasste Netzwerk zu trainieren. Die letztendliche Performance des Modells kann dann mithilfe des Validierungs- bzw. Testsets evaluiert werden.

Viele der verwendeten Netzwerkarchitekturen verlangen, dass die zugeführten Daten, die zum Training verwendet werden sollen, einem vordefinierten Format entsprechen. Aufgrund der Art und Weise, wie die Vektoren und Matrizen innerhalb des Netzwerks verrechnet werden, wird für Inputdaten verlangt, einem bestimmten Format zu entsprechen.

Aus diesem Grund war es notwendig, die zum Training verwendeten Daten in dieses Format zu überführen, um sie für das Training überhaupt brauchbar zu machen. Zum anderen mussten vorliegenden Datensätze weiter transformiert werden, um den Anforderungen einiger Anwendungsfälle zu entsprechen. Neben der angesprochenen Formatierung der Daten mussten diese ebenfalls zurechtgeschnitten und daraufhin eine Skalierung der jeweiligen Datensätze vorgenommen werden. Beispielsweise war es für einige Anwendungsfälle notwendig, dass das Gesicht einer Person im Zentrum des Bildes liegt, gleichzeitig das ursprüngliche Format des Bildes allerdings beibehalten werden sollte. Hierzu wurden Hilfsmethoden erstellt, welche genau diese Aufgaben erfüllen sollten. Die Abbildung 14 zeigt ein Beispiel solcher Hilfsmethoden.

```python
def __detect_faces(image):
    # Create a face detector
    face_detector = dlib.get_frontal_face_detector()

    # Run detector and get bounding boxes of the faces on image.
    detected_faces = face_detector(image, 1)
    face_frames = [(x.left(), x.top(),
                    x.right(), x.bottom()) for x in detected_faces]

    return face_frames


def crop_image(image):
    #scale_percent = 50  # percent of original size
    width = int(image.shape[1])
    height = int(image.shape[0])
    dim = (width, height)

    # resize image
    resized_image = cv2.resize(image, dim, interpolation=cv2.INTER_AREA)

    # Detect faces
    detected_faces = __detect_faces(resized_image)

    # Crop faces and plot
    face = None
    for n, face_rect in enumerate(detected_faces):
        left, top, right, bottom = face_rect
        h = bottom - top + 1
        w = right - left + 1
        left_margin = max(int(left - w * MARGIN), 0)
        top_margin = max(int(top - h * MARGIN), 0)
        right_margin = min(int(right + w * MARGIN), width - 1)
        bottom_margin = min(int(bottom + h * MARGIN), height - 1)

        # changed top-margin to get the forehead into the frame
        face_rect_margin = (left_margin, top_margin - bias_top, right_margin, bottom_margin)
        face = Image.fromarray(resized_image).crop(face_rect_margin)

    if face is None:
        face = np.zeros([224,224],dtype=np.uint8)
        face.fill(255)
        face = Image.fromarray(face)
        left_margin=top_margin=0
        right_margin=bottom_margin = 1

    # Resize cropped face
    cropped_image = face.resize((BASE_WIDTH, BASE_HEIGHT), Image.ANTIALIAS)
    return np.asanyarray(cropped_image)#, left_margin, top_margin, right_margin, bottom_margin
```

Abbildung 14: Beispielausschnitt für die Nutzung von Hilfsmethoden

Das Zurechtscheiden und Skalieren der Datensätze wurde mithilfe des Toolkits dlib[5] ermöglicht. Die „detect_faces“-Methode dient dazu, ein Gesicht auf einem vorliegenden Bild zu erkennen und dieses dann mit einem Rahmen zu versehen, welches dem ursprünglichen Format des Bildes entspricht. Mithilfe der „crop_image“-Methode konnte nun auf Grundlage der zuvor gewonnenen Informationen das Bild so zurechtgeschnitten werden, dass sowohl die zuvor definierte Anforderung des Formates erreicht wurde, als auch das jeweilige Gesicht den Mittelpunkt des jeweiligen Bildes einnahm. Zwar wurde somit die Einhaltung des ursprünglichen Formates des Bildes erfüllt, allerdings war das Problem des für die verwendete Netzwerkarchitektur unpassenden Formates noch nicht behoben. Dieses Problem konnte aber parallel mit der Lösung eines anderen Problems behoben werden, welches im Folgenden vorgestellt werden soll.

In einigen bearbeiteten Machine-Learning-Projekten kam es vor, dass Datensätze mit mehr als 50.000 Bildern verwendet wurden. Wo es beim Laden von Datensätzen mit Stückzahlen im niedrigen vierstelligen Bereich noch zu keinen Problemen kam, führte das gleichzeitige Laden der Bilder bei Datensätzen in höheren Bereichen zu kritischen In-Memory-Errors. Wurden bspw. 20.000 Bilder gleichzeitig in jeweiligen Trainingsprozess geladen, ist der Arbeitsspeicher der verwendeten Maschine übergelaufen und das Training abgebrochen worden. Um dieses Problem zu umgehen, musste ein Weg gefunden werden, die verwendeten Daten stückweise in den Trainingsprozess zu laden, um so eine Überlastung des Arbeitsspeichers zu verhindern. In der in Abbildung 15 dargestellten „load_batch“-Methode wird sowohl das stückweise Laden der Trainings- sowie Testdaten ermöglicht als auch das bestehende Problem des unpassenden Formates behoben.

[5] dlib-Repository auf GitHub. URL: https://github.com/davisking/dlib Abrufdatum: 23.04.2019

```python
def load_batch(images, image_labels, batch_size):

    n = 0
    min_size = 224
    number_images = len(images)
    keypoints=[]

    while 1:

        image_batch = []
        landmark_batch = []

        batch_image = images[n:n+batch_size]
        batch_landmarks = image_labels[n:n+batch_size]

        for i in range(len(batch_landmarks)):

            #resize image
            im = colorImages[:,:,:, frameInd]
            old_size = im.shape[:2]
            ratio = float(min_size)/max(old_size)
            new_size = tuple([int(x*ratio) for x in old_size])
            im = cv2.resize(im, (new_size[1], new_size[0]))
            delta_w = min_size - new_size[1]
            delta_h = min_size - new_size[0]
            top, bottom = delta_h//2, delta_h-(delta_h//2)
            left, right = delta_w//2, delta_w-(delta_w//2)
            color = [255, 255, 255]
            new_im = cv2.copyMakeBorder(im, top, bottom, left, right, cv2.BORDER_CONSTANT, value=color)
            new_im = cv2.cvtColor(new_im, cv2.COLOR_BGR2RGB)

            #resscale landmark coordinates
            jawCords_X = landmarks2D[:16,0,frameInd]
            jawCords_Y = landmarks2D[:16,1,frameInd]

            scale_x = im.shape[0] / colorImages.shape[0]
            scale_y = im.shape[1] / colorImages.shape[1]

            jawMarksRescaled_X = [np.multiply(x, scale_x) for x in jawCords_X]
            jawMarksRescaled_Y = [np.multiply(x, scale_y) for x in jawCords_Y]

            jawMarksRescaled_X = np.round(jawMarksRescaled_X,0)
            jawMarksRescaled_Y = np.round(jawMarksRescaled_Y,0)

            jawMarksRescaled_X = jawMarksRescaled_X + delta_w/2
            jawMarksRescaled_Y = jawMarksRescaled_Y + delta_h/2

            jawMarksRescaled_X_HOLDER = jawMarksRescaled_X.tolist()
            jawMarksRescaled_Y_HOLDER = jawMarksRescaled_Y.tolist()

            keypoints = list(zip(jawMarksRescaled_X_HOLDER, jawMarksRescaled_Y_HOLDER))

            keypoints = batch_landmarks[i]
            img = batch_image[i]
            img = np.array(img)
            img = img[:,:,::-1]
            img = img / 255

            image_batch.append(img)
            landmark_batch.append(keypoints)

            assert len(image_batch) == len(landmark_batch),[len(image_batch),len(landmark_batch)]

            n+=batch_size
            if n >= number_images - batch_size and n < number_images:
                n = number_images-batch_size
            elif n >= number_images:
                n = 0

            yield np.array(image_batch),np.array(landmark_batch)
```

Abbildung 15: „load-batch“-Methode

Die Funktion bekommt sowohl die Trainings- bzw. Testdaten inklusive zugehöriger Labelinformationen bzw. Annotationen der Datensätze, als auch eine zuvor definierte Batchsize

übergeben. Anschließend passiert nichts anders, als dass ein Stück des verwendeten Datensatzes verarbeitet wird und dem Netzwerk zum Training übergeben wird. Wo genau die Methode zum Einsatz kommt, wird im späteren Verlauf dieses Kapitels nochmals genau aufgezeigt. Während dieses Prozesses geschieht im Übrigen auch die Formatierung der in dem Batch enthaltenden Bilder, um das Problem des unpassenden Formats zu beheben. Mithilfe der CV2-Bibliothek werden die entsprechenden Bilder in das angestrebte Format umgewandelt, in diesem Falle 224*224, und anschließend dem zu übergebenen Batch angefügt. Da es sich in diesem Falle um eine „load-batch“-Methode für einen landmarkbasierten Anwendungsfall handelt, müssen die Annotationen der Landmarkkoordinaten ebenfalls entsprechend der zuvor abgeschlossenen Formatierung des jeweiligen Bildes angepasst werden. An dieser Stelle ist ebenfalls anzumerken, dass Hilfsmethoden wie die „load-batch“-Methode je nach Anwendungsfall stets angepasst und adaptiert werden mussten, weshalb die Grundstruktur der Methode zwar konstant ist, allerdings immer wieder kleine Details angepasst werden mussten, um dem vorliegenden Anwendungsfall zu entsprechen.

Das batchweise Laden der Datensätze fand vor allem dann Anwendung, wenn mit großen Datensets gearbeitet wurde. Im Kontrast dazu steht die Herangehensweise an den jeweiligen Anwendungsfall, wenn mit sehr kleinen Datensets gearbeitet wird. Im Falle einer sehr geringen Datengrundlage mussten Verfahren entwickelt werden, den vorhandenen Datensatz künstlich zu strecken mithilfe sogenannter Data Augmentation. Data Augmentation dient dazu, jedes Element des Datensatzes in jeder Trainingsiteration mithilfe verschiedener Verfahren leicht abzuändern, damit das Netzwerk nicht in jeder Periode das exakt gleiche Bild sieht und eine künstliche Disjunktion der Datensätze erzwingt. Die Bibliothek „imgaug“, welche zur Anwendung von Data Augmentation zum Einsatz kam, bot genau die richtigen Funktionen für vorliegende Anwendungsfälle, in welchen aufgrund mangelnder Datenverfügbarkeit nur bedingt gute Ergebnisse erzielt werden konnten. In der gezeigten Methode in Abbildung 16 wird der Ablauf der Augmentation aufgezeigt.

```
def augment(imgs):
    sometimes = lambda aug: iaa.Sometimes(0.5, aug)

    aug_seq_img = iaa.Sequential([
        iaa.Fliplr(0.5),
        sometimes(iaa.Affine(scale=(0.75, 1.25), mode="constant", cval=255)),
        sometimes(iaa.Affine(rotate=(-15,15), mode="constant", cval=255)),
        sometimes(iaa.GaussianBlur((0, 1.0))),
        #sometimes(iaa.AdditiveGaussianNoise(loc=0, scale=(0.0, 0.05*255))),
        #sometimes(iaa.Multiply((0.8, 1.2))),
        iaa.SomeOf((0, 2),[
        #iaa.Sharpen(alpha=(0, 1.0), lightness=(0.8, 1.2)),
        iaa.SimplexNoiseAlpha(iaa.OneOf([
            #iaa.EdgeDetect(alpha=(0.75, 1.0)),
            #iaa.DirectedEdgeDetect(alpha=(0.75, 1.0), direction=(0.0, 1.0)),
        ])),
        iaa.OneOf([
            #iaa.Dropout((0.01, 0.05)),  # randomly remove up to 5% of the pixels
            #iaa.CoarseDropout((0.05, 0.1), size_percent=(0.1, 0.25)),
        ]),

        #iaa.Invert(0.05, per_channel=True),
        iaa.ContrastNormalization((0.75, 1.5)),

        iaa.CropAndPad(
            percent=(-0.0, 0.0),
            pad_mode="constant",
            pad_cval=255
        ),

        ],
        random_order = True
    ),

    ],random_order=True
    )

    images_aug = aug_seq_img.augment_images(imgs)

    return images_aug
```

Abbildung 16: „augment“-Methode

Zum Nachvollziehen der genauen Funktionsweise der verschiedenen Befehle kann in der Dokumentation der Bibliothek[6] mehr gefunden werden.

Das Ergebnis einer Datenaugmentation kann in der Abbildung 17 eingesehen werden. Hierbei handelt es sich nur um eine eher leichte Augmentation des Bildes.

[6]Dokumentation von imgaug. URL: https://imgaug.readthedocs.io/en/latest/index.html Abrufdatum: 23.04.2019

Abbildung 17: Beispiel einer Datenaugmentation

Das Bild ist lediglich etwas gedreht worden, des Weiteren sind einige Randregionen des Bildes weiß eingefärbt worden. Allerdings bietet die Bibliothek diverse Funktionen, um bspw. das Bild zu verzerren, Farbtöne zu ändern oder weich zu zeichnen. Beispiele können in dem imgaug-Repository[7] von Alexander B. Jung auf GitHub eingesehen werden.

Wie bereits in den Imports der Keras-Bibliothek angedeutet, wurde zum Trainieren der Modelle auf vordefinierte Netzwerkarchitekturen zurückgegriffen. In der Übersicht der Machine-Learning-Komponenten wurde auch bereits die Disziplin des Transfer Learnings genannt und wofür es eingesetzt wird. An dieser Stelle soll an einem Beispiel (Abbildung 18) veranschaulicht werden, wie mithilfe der Functional-API von Keras ein vortrainiertes Netzwerk mithilfe von Transfer Learning auf den eigenen Anwendungsfall angepasst wird.

```
inception = InceptionV3(include_top=False, input_shape=(224,224,3))

x = inception.output
x = GlobalAveragePooling2D()(x)
x = Dense(256,activation='relu')(x)
x = Dense(256,activation='relu')(x)
out_landmark = Dense(32, activation='linear', name='landmark')(x)
model = Model(inputs=inception.input, outputs=out_landmark)
model.summary()
```

Abbildung 18: Anpassung eines vortrainierten Netzwerkes mittel Transfer Learning

Die Aufnahme in Abbildung 18 zeigt zunächst das Laden der Inception-Architektur mit entsprechender Input Shape. Anschließend wird der Inception-Architektur sowohl eine Pooling-Layer, als auch zwei Dense-Layer mit je 256 Input-Units und einer ReLu-Aktivierung angefügt. Als Output-Layer wurde eine weitere Dense-Layer hinzugefügt. Sie enthält lediglich 32 Units, da im vorliegendem Landmark-Anwendungsfall sechzehn Landmarks mit je einer X- und Y-Koordinate vorhergesagt werden sollten. Mithilfe von Transfer Learning kann von den bereits hinterlegten Informationen und Features in den jeweili-

[7] imgaug-Repository auf GitHub. URL: https://github.com/aleju/imgaug Abrufdatum: 23.04.2019

gen Gewichten des Netzwerks bei der Bearbeitung eines neuen Anwendungsfalls profitiert werden. Das Verwenden der gespeicherten Gewichte der Inception Architektur wird bei der Deklarierung der „model“-Variable in der oben gezeigten Aufnahme verdeutlicht.

In Kombination mit der zuvor beschriebenen „load_batch“-Methode und dem „model-fit_generator“ von Keras kann das batchweise Laden der Trainings- bzw. Testdaten in dem Trainingsprozess vollzogen werden. Folgende Aufnahme, Abbildung 19, beschreibt diesen Vorgang.

```
model.fit_generator(load_batch(x_train, y_train,batch_size),
                    validation_data=load_batch(x_test, y_test,batch_size),
                    steps_per_epoch=steps,
                    validation_steps=val_steps,
                    epochs=epochs,
                    callbacks=[tensorboard],
                    shuffle=True)
```

Abbildung 19: Vorgang des batchweisen Ladens von Trainings- bzw. Testdaten

Hier kann ebenfalls der Aufruf der von Keras bereitgestellten Callback-Funktion in Kombination mit Tensorbaord betrachtet werden.

9.5 Anforderungen

Nachdem das allgemeine Vorgehen und die zur Verfügung stehenden Werkzeuge zur Bearbeitung von Anwendungsfällen beschrieben wurden, kann sich nun der Anforderungserhebung gewidmet werden. Wie bereits erläutert, orientiert sich das allgemeine Vorgehen am Modell des CRISP-DM, allerdings in adaptierter Art. Bevor dieses jedoch angewendet werden kann, muss zunächst eine Grundlage geschaffen werden, auf die das Vorgehensmodell angewandt werden kann. Diese Grundlage stellt die Anforderungserhebung dar.
Zunächst ein paar allgemeingültige Informationen. Alle erhobenen Anforderungen wurden in enger Zusammenarbeit mit dem Praxispartner Brille24 ausgearbeitet. Vor allem Mitarbeiter der Fashion- und Designabteilung von Brille24 waren in diesem Prozess maßgeblich involviert. Der Betreuer und Ansprechpartner bei Brille24 war vor Allem daran beteiligt, die Projektgruppe bezüglich der Findung entsprechender Anforderungen auf die richtige Spur zu bringen. Des Weiteren fungierte er als Koordinator und Bindeglied zwischen der Projektgruppe und den verschiedenen Abteilungen von Brille24. Für alle identifizierten Merkmale und daraus resultierenden Anforderungen sind diese zuvor genannten Informationen allgemeingültig und werden deshalb nicht weiter in der näheren Beschreibung der jeweiligen Anforderung erwähnt. Jede erhobene Anforderung wird zunächst kurz beschrieben und anschließend in den Projektkontext gesetzt. Somit soll die entsprechende

Notwendigkeit der jeweiligen Anforderung verdeutlicht werden.

Die Frage, die es im Kontext der Anforderungserhebung zu beantworten gab, war, welche spezifischen Merkmale und Features eines betrachteten Gesichts relevant sind, um fundierte Produktvorschläge zu generieren. Spezifischer galt es zu berücksichtigen, welche modischen Implikationen vom jeweils betrachteten Merkmal resultieren und wie bedeutend dessen Einfluss auf den potentiellen Brillenvorschlag ist. Im Verlauf des Projekts übermittelte der Praxispartner Brille24 zwar einen Regelkatalog, um die Relevanz eines extrahierten Features bewerten zu können und anschließend auf entsprechende Produkte zu mappen. Dieser stand in der Anfangsphase des Projektes allerdings noch nicht zur Verfügung. Aus diesem Grund basiert die Ausarbeitung der Anforderungserhebung lediglich auf der Zusammenarbeit mit Brille24, ohne Rücksicht auf die später formulierten Regeln zum Ableiten der Featurerelevanz. Bezüglich des Brillenberaters wurden Merkmale identifiziert, um fundierte Produktvorschläge zu ermöglichen, welche im Folgenden kurz aufgelistet werden:

- Geschlecht der betrachteten Person
- Alter der betrachteten Person
- Haar – und Hautfarbe der betrachteten Person
- Trägt die Person einen Bart
- Gesichtsform der betrachteten Person

An dieser Stelle ist bereits anzumerken, dass sich das Merkmal der Gesichtsform nochmals untergliedern lässt in verschiedene Arbeitspakete, dazu allerdings später mehr. Diese zuvor genannten Merkmale stellen die Grundlage dazu dar, das Gesicht einer betrachteten Person in verschieden Komponenten zu unterteilen, diese zu bewerten und anschließend auf einen jeweiligen Produktvorschlag zu übertragen. Was nun folgt, ist die Umformulierung dieser Merkmale in jeweilige Anforderungen.

Klassifizierung des Geschlechts

Die erste Anforderung des Brillenberaters, die erfüllt werde muss, ist die Klassifikation des Geschlechtes der jeweilig betrachteten Person. Männer und Frauen tragen unterschiedliche Mode, dies verhält sich bei Brillen nicht anders. Um ein Beispiel zu nennen, Brillenmodelle der Cateye-Gruppe sind vorzugsweise weiblichen Kunden zu unterbreiten und sind im Allgemeinen als reine Frauenbrille angesehen. Zwar können auch Männer dieses Brillenmodell

tragen, allerdings sollte dies unter Anbetracht modischer Aspekte jedoch vermieden werden. Deshalb ist es für den Brillenberater notwendig, diese Unterscheidung durchzuführen, da dies letztendlich als entscheidendes Filterungskriterium für potenzielle Brillenvorschläge dient.
Die allgemeine Anforderung der Klassifizierung des Geschlechts wird außerdem umfasst von der Anforderung, eine jeweilige Performance zu erfüllen. Bei der Evaluierung trainierter Machine Learning Modelle muss eine entsprechende Metrik gewählt werden, um die Performance des Netzes evaluieren zu können. Im Allgemeinen wurde die Anforderung an das jeweilige Netz zur Klassifizierung des Geschlechts gesetzt, eine Klassifizierungsgenauigkeit von ca. 98% zu erfüllen. Genauigkeit in diesem Falle bedeutet, dass bei der Bewertung der Performance die Gesamtanzahl aller korrekten Prognosen geteilt durch die Summe aller gemachten Prognosen nicht kleiner als 0,98 sein darf. Diese Anforderung resultiert aus der Erkenntnis der Sensibilität des Kunden gegenüber Beratungsservices. Betrachtet man, rein hypothetisch, einen Klassifizierungsalgorithmus, der ebenso die Klassifizierung des Geschlechts einer betrachteten Person durchführt, aber statt einer Klassifizierungsgenauigkeit von 98% nur eine Genauigkeit von ca. 60% aufweist. Im Schnitt würde das Geschlecht eines jeden dritten Kunden falsch klassifiziert werden, was letztendlich in potenziell falschen Produktvorschlägen bei jeden dritten Kunden resultieren würde. Kunden würden daraufhin unwiderruflich den Service wechseln. Dies ist keineswegs tragbar, weshalb eine möglichst hohe Klassifizierungsgenauigkeit innerhalb des Service obligatorisch ist. Generell lassen sich diese Regeln und Performanceanforderungen auch auf die noch folgenden Anforderungen übertragen.

Bestimmung des Alters der betrachteten Person
Ebenso wie die zuvor beschrieben Anforderung der Klassifizierung des Geschlechts ist auch die Bestimmung des Alters der betrachteten Person eine Notwendigkeit, um in Anbetracht modischer Aspekte fundierte Produktvorschläge zu generieren. Unterschiedliche Altersgruppen tragen unterschiedliche Brillenmodelle. Dies muss bei potenziellen Produktvorschlägen berücksichtigt werden. Das geschätzte Alter der jeweiligen Person stellt dementsprechend ein weiteres Filterkriterium dar, an dem der Raum potenziell vorgeschlagener Brillen weiter eingegrenzt werden kann.
Ebenso wie bei der Anforderung zu Performance bzw. der Genauigkeit der Prognosen besteht auch bei der Bestimmung des Alters der betrachteten Person die Anforderung, eine möglichst präzise Prognose zu treffen. Als Anforderung wurde somit definiert, dass das vom Netzwerk prognostizierte Alter der jeweiligen Person nicht um mehr als fünf Jahre vom eigentlichen Alter abweichen darf.

Klassifizierung von Bartträgern
Ein weiteres Feature, welches bei der Brillenberatung in den Filterprozess mit einfließt, ist die Tatsache, ob die betrachtete Person einen Bart besitzt oder nicht. Unter modischen Aspekten betrachtet kann ein Bart das generelle Aussehen und die Struktur des Gesichts maßgeblich verändern. Deshalb wurde der Bart bzw. Nicht-Bart als weiteres Anforderungskriterium definiert, da er, ebenso wie die zuvor genannten Anforderungen, ebenfalls den Filterungsprozess bei der Generierung kundenspezifischer Brillenvorschläge beeinflusst.
Auch hier wird zusätzlich eine Anforderung an die Performance der Klassifizierungsgenauigkeit der Prognosen definiert. Diese solle ebenfalls eine Klassifizierungsgenauigkeit von ca. 98% aufweisen.

Klassifizierung der Haar- und Hautfarbe
Ein weiteres Merkmal, welches vor allem ausschlaggebend für die Wahl der Farbe des Brillenmodells bei jeweiliger Prognose ist, wären die Features der Haar- und Hautfarbe. Dies ist vor allem deshalb notwendig, da bspw. Personen mit sehr hellem Hauttyp keine Brille tragen sollten, welche ebenfalls weiß, durchsichtig oder ähnliches ist. Durch die Klassifizierung des entsprechenden Farbtyps jeweiliger Features können oben genannte Kombinationen aus Haut- und Brillenfarbe verhindert werden. Somit stellt auch diese Anforderung einen weiteren Filterungsprozess dar, mithilfe dessen der Raum an Brillenvorschlägen weiter eingegrenzt werden kann.
Die jeweilige Performance des Netzwerkes bezüglich der Genauigkeit der Prognosen ist auch hier eine weitere zentrale Anforderung, die definiert wurde.

Bestimmung der Gesichtsform
Eine der zentralen Anforderungen innerhalb des Brillenberaters stellt die Erkennung der Gesichtsform in dem aufgenommenen Foto dar. In Gesprächen mit der Fashionexpertin des Praxispartners Brille24 kristallisierte sich heraus, dass dieses Feature essenziell für eine individuelle Produktempfehlung ist. So gelten beispielsweise eindeutige Style-Richtlinien, etwa dass Personen mit einem runden Gesicht aus einer Fashionperspektive betrachtet keine runden Brillengläser tragen sollten. Vielmehr sollten sie zu einer eckigen Brillenform greifen, da eine solche als Kontrast zum runden Gesicht dem eigenen Antlitz in einem solchen Falle eher schmeichelt. Die Wichtigkeit des Features „Gesichtsform“ manifestiert sich konsequenterweise auch darin, dass dieses den größten Einfluss innerhalb des Regelkatalogs hat, welcher von Brille24 erstellt und zur Verfügung gestellt wurde und eine Kombination von Gesichtsfeatures zu einem individuellen Brillenvorschlag mappt. Der hier angedeutete

Vorgang des Mappings wird im weiteren Verlauf der Dokumentation noch ausführlicher beschrieben. Die Gesichtsform soll entsprechend des zuvor erwähnten Brille24-Regelkataloges in fünf Gruppen unterteilt werden, namentlich:

- Eckiges Gesicht
- Herzförmiges Gesicht
- Ovales Gesicht
- Rundes Gesicht
- Trapezförmiges Gesicht.

Zur Bestimmung der Gesichtsform können zwei Ansätze unterschieden werden: Der erste Ansatz basiert auf sogenannten Landmarks, welche charakteristische Punkte in einem Gesicht beschreiben wie zum Beispiel Nasenspitze, Kinn etc.. Der zweite Ansatz ist eine direkte Klassifikation mithilfe entsprechend gelabelter Bilder. Auf Basis dieser beiden Ansätze können zugehörige neue Unteranforderungen erhoben werden.

1. **Bestimmung von Facial Landmarks**
   Die Bestimmung von Facial Landmarks als Zwischenschritt zur Ermittlung der Gesichtsform basiert auf Wünschen des Praxispartners Brille24. Dieser stellte in Aussicht, dass ein Industriepartner im Verlauf der Projektzeit Regeln bereitstellen könne, welche aus den Landmarks Gesichtsformen ableiten, beispielsweise auf Grundlage von errechneten Distanzmaßen wie der Länge des Gesichts (z. B. als Distanz der y-Koordinate der Kinnspitze zu der y-Koordinate des Stirnzentrums). Im Detail sollen, wie in Abbildung 20 dargestellt, 76 Landmarks bestimmt werden. Diese Zahl ergibt sich zum einen Teil aus den 68 klassischen Landmarks, welche etwa von der Facial Landmark Detection aus dlib ermittelt werden. Die übrigen acht Landmarks wurden in Absprache mit Brille24 ausgewählt, um fehlende Gesichtsregionen bzw. -merkmale abzudecken. Sechs dieser Landmarks betreffen die Stirnregion, die zwei übrigen befinden sich am Nasenrücken und können in der Theorie hilfreich sein, um zu ermitteln, ob der Nasensteg einer Brille passgenau sitzt.

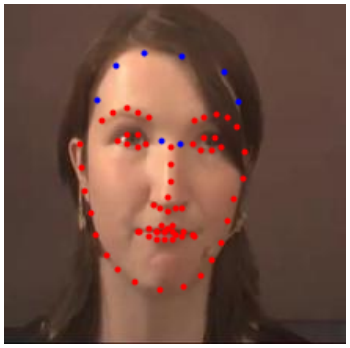


Abbildung 20: Darstellung der 76 Landmarks bestehend aus 68 vordefinierten Landmarks (rot) und 8 selbst gelabelten Landmarks (blau)

2. **Klassifizierung der Gesichtsform**
   Ein weiterer möglicher Ansatz zur Bestimmung der Gesichtsform besteht darin, diese direkt mithilfe eines entsprechend gelabelten Datensets zu klassifizieren. Da die Gesichtsform kein verhältnismäßig einfach und offensichtlich zu bestimmendes Feature ist, wie etwa das Geschlecht, stellt das Auffinden eines solchen Datensets eine Herausforderung dar. Entsprechend des eingangs erwähnten Regelkatalogs zum Mapping der Gesichtsmerkmale zu Produktvorschlägen sollte die Gesichtsform idealerweise in die Ausprägungen eckiges, herzförmiges, ovales, rundes sowie trapezförmiges Gesicht klassifiziert werden. Diese Anforderung wurde von der Projektgruppe als Alternative zum Ansatz der Facial Landmarks als Zwischenschritt betrachtet. Aufgrund der im Vergleich erhöhten Simplizität der direkten Klassifizierung war diese Herangehensweise die eigentlich präferierte. Als Problem sollte sich jedoch die fehlende Datengrundlage herausstellten, sodass die Anforderung erst in der Endphase der Projektgruppe umgesetzt werden konnte.

Mapping der Merkmalen zu den Brillen

Anhand der bis hierhin formulierten Anforderungen ist es nur möglich, den Kunden zu beschreiben. Die Zielsetzung der Beratung ist damit noch nicht erfüllt, da der Kunde alleine aus den Merkmalen keine Rückschlüsse auf passende Produkte ziehen kann. Dementsprechend muss ein Mapping Verfahren entwickelt werden, welches dem Kunden auf Basis der extrahierten Merkmale passende Brillen auswählt. Die Zielsetzung dieser Anforderung besteht darin, dass ein vorgegebener Produktkatalog auf wenige Brillen reduziert werden kann, welche dem Kunden dann vorgeschlagen werden können.

Eine Übersicht der gesammelten Anforderungen für den Brillenberater ist in Tabelle 17 abgebildet.

Nr.	Beschreibung
BB.1	**Klassifizierung des Geschlechts**
BB.2	**Bestimmung des Alters der betrachteten Person**
BB.3	**Klassifizierung von Bartträgern**
BB.4	**Klassifizierung von Haar- und Hautfarbe**
BB.5	**Bestimmung der Gesichtsform**
BB.5.1	**Bestimmung von Facial Landmarks**
BB.5.2	**Klassifizierung der Gesichtsform**
BB.6	**Mapping der Merkmalen zu den Brillen**

Tabelle 17: Anforderungen an Brillenberater

9.6 Umsetzung

Nachdem alle Anforderungen an die Machine Learning Komponenten innerhalb des Brillenberaters aufgeführt und beschrieben wurden, kann sich nun dem allgemeinen Vorgehen bei der Bearbeitung jeweiliger Arbeitspakete gewidmet werden. Im Folgenden soll in Reihenfolge der genannten Anforderungen das Vorgehen bei der Bearbeitung der Teilprojekte geschildert werden.

9.6.1 Extraktion der Merkmale

Im ersten Abschnitt der Umsetzung werden die Arbeitsschritte beschrieben, die notwendig waren, um die relevanten Merkmale aus den Kundenbildern zu extrahieren.

BB.1 Klassifizierung des Geschlechts Basis eines jeden Machine Learning Anwendungsfalles stellt das Vorhandensein einer spezifischen, auf den Anwendungsfall angepassten Datengrundlage dar. Um den zuvor definierten Anforderungen gerecht zu werden, musste sich deshalb zunächst einmal spezifisches Domänenwissen angeeignet werden. Wie bereits in der Beschreibung des allgemeinen Vorgehens in Machine Learning Projekten beschrieben, wurde sich dabei stark an dem adaptierten CRISP-DM-Zyklus orientiert. Dementsprechend müssen vor allem passende Datenquellen identifiziert werden.

Im Internet existieren viele unterschiedliche Ressourcen und Quellen, um vorgefertigte

Datensets zu beziehen. Diese sind zumeist Open Source und für wissenschaftliche Arbeit und Forschung verwendbar. Zwar existieren eine Vielzahl dieser Quellen, allerdings besteht die Herausforderung darin, passende und auf den Anwendungsfall zugeschnittene Datenquellen zu identifizieren. Betrachtet man den Kontext, in welchem der Anwendungsfall der Klassifizierung des Geschlechts der jeweiligen Person erfüllt werden muss, können spezifische Kriterien herausgefiltert werden, die von der potenziellen Datenquelle erfüllt werden müssen. Im Falle des Geschlechts wäre dies ein Datensatz, in dem beide Geschlechtsgruppen in möglichst ausgeglichener und hoher Anzahl vorliegen. Aufgrund der Rahmenbedingungen, in denen das betrachtete Machine Learning Modell angewendet wird, wird außerdem nach einem Datenset mit Bildern gesucht, in denen die jeweilige Person möglichst zentriert ist und möglichst nur das Gesicht sowie der Schulterbereich der Person zu sehen ist. Dies ist nötig, da bei der Verwendung des Brillenberaters durch den Nutzer ebenfalls ein zentriertes Foto mithilfe der Frontkamera des jeweiligen Endgerätes übermittelt wird. Somit muss das Machine Learning Modell ebenfalls lediglich auf diese Features achten. Ein Datenset, welches diesen Anforderungen entspricht, ist das Appa-Real-Dataset und das IMDB-WIKI-Dataset. Allgemeine Informationen zu den Eigenschaften des Sets sind in der Übersicht der Komponenten zu finden. Nachdem das passende Datenset identifiziert wurde, kann der eigentliche Prozess des Preprocessings bis hin zum Training des Modells beginnen.

Zur Entwicklung der Modelle wurde auf Jupyter Notebooks zurückgegriffen. Neben den Imports aus benötigten Frameworks und Libraries, werden des Weiteren entsprechende Methoden verwendet, die das eigentliche Laden der jeweiligen Daten ermöglichen. Im Falle des IMDB-WIKI-Datasets mussten ca. 500.000 Bilder für das Training geladen werden. Um diese Menge verarbeiten zu können, musste ein Chunking vorgenommen werden, sodass die Datensätze Stück für Stück in das Training geladen werden. Aus diesem Grund mussten ebenfalls Methoden entwickelt werden, die die jeweiligen Datensätze batchweise laden, damit potenzielle Memory-Errors, verursacht durch das überladen des Arbeitsspeichers, vermieden werden können. Nachdem das Laden sowie batchweise Laden ermöglicht wurde, konnten weitere Schritte vorgenommen werden, um den Datensatz auf das Training vorzubereiten. Ein weiterer elementarer Schritt ist es, die verwendeten Daten in einen passenden Zustand zu bringen. Zustand bedeutet in diesem Falle, dass die Bilddateien auf eine passende Größe zurechtgeschnitten werden, damit sie effektiv vom Netzwerk verarbeitet werden können. Einige Netzwerkarchitekturen verlangen beispielsweise, dass übergebene Bilder einer gesetzten Größe bzw. einem festgelegten Format entsprechen. In der Forschung wurde die Konvention übernommen, dass bei Anwendungsfällen im Bereich Computer Vi-

sion das Bildformat bei 222*224 liegt. Anschließend folgt das letztendliche Aufteilen des Datensets in Trainings- sowie Testdatenset. In der Machine Learning Domäne hat sich die Aufteilung des Datensets in Trainings- sowie Testset als Standard herausgestellt. Viele Projekte gehen auch noch einen Schritt weiter und trennen das vorliegende Datenset in Trainings-, Test- und Validierungsset, um bspw. das Validierungsset mithilfe von Cross-Validation zu bewerten und eine Abschließende Evaluation der Performance am Testset durchzuführen. Im Rahmen des Projektes wurde allerdings nur auf die Aufteilung des Datensets auf Trainings-, sowie Validierungsset zurückgegriffen.

Keras bietet eine Vielzahl vorimplementierter Netzwerkarchitekturen. Zwar wurde anfangs gesagt, dass zu Beginn auf ein eigens konstruiertes Custom-Netzwerk zurückgegriffen wurde, allerdings war die Performance nicht annähernd an den von uns und dem Praxispartner definierten Threshold von 98% Accuracy, welcher im Rahmen der Anforderungserhebung definiert wurde. Deshalb wird an dieser Stelle auf die weitere Beschreibung dieser Herangehensweise verzichtet. Anzumerken ist jedoch, dass das Team verschiedene Gründe identifiziert hat, die zu der letztendlich ernüchternden Performance geführt haben könnten. Hauptgrund schien die unzureichende Tiefe des Netzwerkes gewesen zu sein. Das Netzwerk war einfach nicht tief und komplex genug, um der Anzahl der komplexen extrahierten Features gerecht zu werden. Diese Annahme konnte dadurch bestätigt werden, dass bei der Verwendung einer vordefinierten, vorimplementierten Netzwerkarchitektur wesentlich bessere Ergebnisse erzielt werden konnten als zuvor. Als Architektur wurde InceptionV3 verwendet, welche bereits in der Komponentenübersicht beschrieben wurde. Keras bietet dabei die Option, die vortrainierten Gewichte des Netzwerkes bei Trainingsbeginn mitzuladen, um so von Transfer Learning zu profitieren, oder das Netzwerk ohne vortrainierte Gewichte zu laden und so alle Features der Bildobjekte von Grund auf neu zu lernen. Die besten Ergebnisse wurden erzielt, als Transfer Learning angewendet wurde. Hierbei müssen lediglich ein paar Änderungen an der Netzwerkarchitektur vorgenommen werden, um sie auf den eigenen Anwendungsfall anzupassen. Im Falle der Klassifizierung des Geschlechts mussten die letzten beiden Layer der InceptionV3-Architektur abgeschnitten werden, um diese durch auf den Anwendungsfall angepasste Layer zu ersetzten. Da es sich bei der Klassifizierung des Geschlechts um eine binäre Klassifikation handelt, wurde eine Dense-Layer mit lediglich 2 Input-Units ans Output-Layer an die Architektur angefügt. Als Aktivierungsfunktion wurde die Sigmoid-Funktion verwendet, da der Output zwischen 0 und 1 liegen sollte. Je stärker der Wert in eine Richtung polarisiert, desto sicherer ist das Netzwerk bei der Vorhersage, ob es sich um eine männliche oder weibliche Person auf dem Bild handelt. Es wurde viel an den Hyperparametern eingestellt und getestet, die

besten Ergebnisse wurden erzielt, als LogCosh als Loss-Funktion und SGD als Optimizer verwendet wurden.

Es stellte sich im Laufe des Projektes jedoch heraus, dass das trainierte Netzwerk zu groß war, um effektiv vom Backend verarbeitet zu werden. Eines der Ziele der im Projekt verwendeten Architektur war es zudem, eine einfache Skalierbarkeit zu ermöglichen. Die Größe des Netzes hätte dies aber nur bedingt zugelassen, weshalb die bisherigen Ergebnisse zum damaligen Stand überarbeitet werden mussten. Ziel war es jetzt nicht mehr nur eine entsprechende Performance zu erreichen, sondern auch noch möglichst geringe Größe (Größe im Sinne von der benötigten Speicherkapazität) zu erzielen. Eine Architektur, die diesen Anforderungen entspricht, ist die MobileNetV2-Architektur. Der zentrale Vorteil von MobileNet ist, dass die Parameteranzahl innerhalb des Netzwerkes deutlich reduziert werden kann, bei gleichzeitig geringen Einbußen in der Performance. In der Übersicht der Netzwerkarchitekturen in der Komponentenübersicht kann dieser Sachverhalt nochmals nachgesehen werden. Durch die stark reduzierte Parameteranzahl wurde auch die Dateigröße ebenfalls wesentlich reduziert und deshalb perfekt auf den Anwendungsfall ausgelegt. Nun wurde ein neues Netzwerk auf Basis von MobileNet anstelle von Inception trainiert, wobei zuvor gesetzte und verwendete Hyperparamter wie Optimizer und Loss-Funktion ebenfalls verändert werden mussten. Die beste Performance konnte erzielt werden, als ADAM mit einer gesetzten Learning Rate von 0,001 und Categorical Crossentropy als Loss-Funktion verwendet wurden. Zwar erzielte diese Kombination von Hyperparametern die beste Performance, allerdings wurde der in den Anforderungen gesetzte Threshold der Klassifizierungsperformance von 98% nicht erreicht. Es war voraussehbar, dass die deutlich reduzierte Parameteranzahl die letztendliche Performance des Netzwerkes verringern würde, jedoch war die erzielte Klassifizierungsgenauigkeit von ca. 89% nicht ausreichend, um das Arbeitspaket abzuschließen und zu deployen. Doch auch für dieses Problem konnte eine effektive Lösung gefunden werden.

Die Lösung bestand darin, dass auf MobileNet trainierte Netzwerk nochmals auf einen neuen, kleineren Datenset mithilfe von Finetuning zu verbessern. Die Idee besteht darin, das trainierte Netzwerk nochmals auf einem reduzierten, aber für das Netzwerk bisher fremden Datenset zu trainieren, um so potentielle Verbesserungen in der Performance zu erzielen. Auch hier handelte es sich um die Methodik des Transfer Learnings. Zum Finetunen des Netzwerkes wurde auf das Datenset Appa-Real-Datenset zurückgegriffen, einem Datenset, welches die Kriterien des Anwendungsfalles erfüllt und zudem Ähnlichkeiten zum IMDB-WIKI-Datensatz aufweist. Eine Verbesserung der Performance konnte ebenfalls durch die

Tatsache erzielt werden, dass sowohl das Appa-Real- als auch das IMDB-WIKI-Datenset nochmals händisch nach Datenqualität kontrolliert wurde und unbrauchbare Bilder sowie falsch annotierte Dateien entfernt bzw. verbessert wurden. Beim Überfliegen der Datensätze ist unter anderem aufgefallen, dass die Qualität der Daten zum Teil stark variiert und dies die Performance des Netzwerks unnötig mindern würde. Das beste Ergebnis, welches mithilfe dieses Verfahrens erzielt werden konnte, war eine Klassifizierungsgenauigkeit von ca. 95%. Zwar wurden die angestrebten 98% nicht erreicht, allerdings hatte das Netzwerk nun eine deutlich reduzierte Parameteranzahl, eine daraus resultierende deutlich reduzierte Dateigröße und konnte nun problemlos vom Backend verarbeitet werden. Eine Reduzierung von ca. 200 MB auf 14MB mit einer Einbuße von ca. 3% in der letztendlichen Performance ist ein Trade-Off, der gerne hingenommen wurde. Damit konnte der Anwendungsfall der Klassifizierung des Geschlechts abgeschlossen werden.

BB.2 Bestimmung des Alters der betrachteten Person Anders als bei der Klassifizierung des Geschlechts handelt es sich bei der Bestimmung des Alters um keine einfache binäre Klassifikation. Bevor mit dem eigentlichen Vorhaben gestartet werden konnte, musste zunächst festgelegt werden, auf welche Art und Weise das Alter überhaupt bestimmt bzw. betrachtet werden soll. Zur Auswahl standen mehrere Herangehensweisen. Ein vielversprechender Ansatz war beispielsweise die Einteilung des Alters in Gruppierungen, sodass man nicht direkt das Alter der jeweiligen Person prognostiziert, sondern lediglich eine Zuordnung der vorliegenden Person einer zuvor definierten Altersgruppe. Diese Gruppen hätten aus Segmenten bestanden wie bspw. Kind, Erwachsener oder Personen im hohen Alter, welche man dann wiederum in weitere Untersegmente hätte aufgliedern können. Ein weiterer Ansatz, der zur Auswahl stand, war die exakte Bestimmung des Alters der jeweiligen Person. Hierbei konnte auch noch darin unterschieden werden, ob das tatsächliche Alter der Person geschätzt werden soll oder lediglich das wahrgenommene Alter. Nach ausführlicher Absprache wurde sich darauf geeinigt, dass das Model das wahrgenommene Alter vorhersagen soll, da dies in Anbetracht der modischen Fokussierung des Dienstes des Brillenberaters am besten passen würde. Für die modische Beratung ist es wichtiger, den tatsächlichen Look des Kunden zu berücksichtigen und auf Grundlage des wahrgenommenen Alters einen Produktvorschlag zu generieren, da lediglich Wert auf das optische Zusammenspiel von Gesicht und Brille gelegt wird.

Glücklicherweise musste nicht so viel Zeit in die Prozesse des Datenverständnisses und der Datensichtung aufgewendet werden, da die grundsätzliche Ausgangslage stark mit der Ausgangslage der Klassifizierung des Geschlechts korrelierte. Ähnlich Willkommen war

ebenfalls die Tatsache, dass in den verwendeten Datensätzen bei der Klassifizierung des Geschlechts bereits Annotationen für das Alter hinterlegt sind. Dementsprechend konnte für die Bestimmung des Alters ebenfalls das IMDB-WIKI- sowie das Appa-Real-Datenset verwendet werden. Entscheidender Unterschied zum zuvor behandelten Anwendungsfall war nun jedoch das letztendliche Ergebnis bzw. der letztendliche Output, der vom jeweiligen Netzwerk zurückgegeben werden sollte. Aus diesem Grund mussten einige Anpassungen gegenüber der verwendeten Architektur zur Klassifizierung des Geschlechts vorgenommen werden. Vorweg: Die Modelle zur Bestimmung des Alters und des Geschlechts sind relativ zeitnah und parallel zueinander entstanden. Somit trat auch innerhalb dieses Arbeitspakets das Problem auf, dass die Größe des Modells verringert werden musste. Deshalb wird an dieser Stelle direkt mit der Verwendung der MobileNet-Architektur fortgeführt, welche zur Lösung des Problems herangezogen wurde. Das Vorgehen, zunächst ein Pre-Training auf dem IMDB-WIKI-Datenset durchzuführen und anschließend ein Fine-Tuning auf dem Appa-Real-Datensatz vorzunehmen, wurde ebenfalls beibehalten. Was verändert werden musste, war die Wahl einer passenden Metrik zur Bewertung der Performance des Netzwerkes während des Trainingsprozesses und Anpassungen der Netzwerkarchitektur bezüglich der Anwendung von Transfer Learning. Im Falle der binären Klassifikation wurden die letzten beiden Layer abgeschnitten und eine Dense-Layer mit zwei Units als Output Layer angehangen. Da das vorliegende Modell nur einen Wert vorhersagen soll, wurde stattdessen eine Dense-Layer mit nur einer Input-Unit angefügt. Des Weiteren hat sich herausgestellt, dass es zu einer Performanceverbesserung kam, nachdem zusätzliche Dense-Layer mit je 256 Units und ReLu als Aktivierungsfunktion der jeweiligen Neuronen zwischen der Output-Layer und dem abgetrennten Teil der bestehenden Netzwerkarchitektur angefügt wurden. Ebenfalls angepasst wurde die Aktivierungsfunktion der angefügten Output-Layer. Da es sich im übertragenen Sinne beim betrachteten Anwendungsfall um ein Regressionsproblem handelt, wurde eine einfache lineare Aktivierungsfunktion gewählt. Zur Bewertung der Performance während des Trainingsprozesses wurde auf den Mean-Absolute-Error zurückgegriffen, um die relative Abweichung der Vorhersage zum tatsächlichen Wert des jeweiligen Objektes zu bestimmen. Mit einer durchschnittlichen Abweichung von ca. 5 Jahren wurde das Training des Netzwerkes eingestellt und abgeschlossen.

BB.3 Klassifizierung von Bartträgern Bei der Klassifizierung, ob eine jeweilige Person einen Bart trägt oder nicht, handelt es sich ähnlich wie bei der Klassifizierung des Geschlechts um eine einfache binäre Klassifikation. Ebenso wie bei der Klassifizierung des Geschlechts soll das Netzwerk einen Output generieren, der in einem Wertebereich zwi-

schen [0,1] liegt, wobei stark polarisierende Werte nahe 0 bzw. 1 signalisieren, dass jeweilige Person einen Bart trägt bzw. keinen Bart trägt.

Das LFW-Datenset beinhaltet genau die passenden Annotationen, um dieses Klassifizierungsproblem zu bearbeiten. LFW besteht aus ca. 13.000 Bildern mit entsprechenden Annotationen. Allerdings hat sich beim händischen Kontrollieren des Datensatzes herausgestellt, dass zum einen teilweise unbrauchbare Bilder enthalten waren und zum anderen die für das entsprechende Bild hinterlegten Annotationen fehlerhaft waren. Dementsprechend könnte sich die Datenqualität des Datensets negativ auf die Performance des Netzes auswirken. Es bestand nun die Möglichkeit, den kompletten Datensatz zu verwenden und den vorhandenen Noise zu ignorieren, das Datenset händisch nochmals nachzubessern oder einen reduzierten Datensatz des LFW-Sets zu verwenden. Hierzu müsste der Datensatz ebenfalls händisch überflogen werden und passende Bilder inklusive Annotationen in einem separaten File abgelegt werden. Nach interner Diskussion und Absprache mit dem Praxispartner wurde sich dazu entschieden, den vorhandenen Datensatz zu reduzieren und so zwar nur mit einem Bruchteil der zur Verfügung stehenden Daten zu arbeiten, dafür aber eine hohe Datenqualität sicherzustellen und zu garantieren. Da sowohl Transfer Learning als auch Image Augmentation angewendet werden soll, könne man so die doch eher geringe Anzahl an Daten kompensieren und dennoch eine zufriedenstellende Performance erzielen.

Der letztendliche Datensatz zum Trainieren des Netzes bestand aus ca. 2500 Daten, welche händisch aus dem LFW-Datensatz extrahiert wurden. Dabei wurde darauf geachtet, dass die jeweilige Anzahl der Bart- bzw. Nicht-Bart-Bilder möglichst ausgeglichen ist, um eine Balance der Daten sicherzustellen. Die Annotationen des LFW-Datensets sind in einem separaten Textfile hinterlegt. Für den betrachteten Anwendungsfall werden sowohl nur die Annotation für Bart/Nicht-Bart benötigt, als auch lediglich die entsprechenden Annotationen für die von uns extrahierten Bilder. Aus diesem Grund mussten ebenfalls Hilfsmethoden erstellt werden, welche diese Informationen aus dem Textfile herauslesen und in einem separaten Dokument speichern, welches anschließend dazu verwendet wird den jeweiligen Bildern die entsprechenden Annotationen zuzuordnen. Anschließend konnte der Datensatz in Trainings- sowie Testsets aufgeteilt werden und mit der Konstruktion des Netzes begonnen werden.

Zunächst eine Anmerkung: In den vorherig beschriebenen Arbeitspaketen wurde auf das Problem eingegangen, dass die trainierten Netze zu groß waren, um von der Architektur verarbeitet zu werden. MobileNet wurde als Architektur herangezogen, um dieses Pro-

blem zu beheben. Das Bartmodell ist jedoch in einer späteren Phase entwickelt worden, in welcher die genannten Kapazitätsprobleme keine Rolle mehr spielten. Aus diesem Grund konnten wieder andere Netzwerkarchitekturen der Keras Bibliothek verwendet werden. Zwar hätte vorliegender Anwendungsfall ebenfalls mit MobileNet gelöst werden können, da jedoch auf einen stark reduzierten Datensatz zurückgegriffen wurde, war Transfer Learning ein entscheidendes Kriterium, um die anvisierte Performance zu erreichen. Deshalb wurde eine Architektur verwendet, die eine deutlich höhere Parameteranzahl aufweist als die von MobileNet, um das Defizit der reduzierten Datenmenge zu umgehen.

Als Basis wurde auf die ResNet50-Architektur zurückgegriffen. Diese Auswahl beruht keineswegs auf Grundlage der jeweiligen Performance, denn wie die Übersichtstabelle der Machine-Learning Komponenten zeigt, existieren Architekturen wie Inception, welche performanter sind als ResNet. ResNet wurde ausgewählt damit nicht in jedem bearbeiteten Anwendungsfall Inception als Architektur eingesetzt wird und da ein großes Interesse bestand diese Architektur anzuwenden. Da auch mit ResNet die zu erzielende Performance erreicht wurde, gab es auch keinen Grund, die Architektur nochmals auszuwechseln. Wie schon in den Anwendungsfällen zuvor wurden erneut die letzten beiden Schichten des Netzes abgetrennt und durch eine Dense-Layer mit je zwei Units ersetzt. Als Aktivierungsfunktion der Output-Layer wurde die Softmax-Funktion verwendet, um ebenfalls eine Ausgabe im Wertebereich [0,1] zu erhalten. Als Optimizer wurde erneut ADAM eingesetzt, die Learning-Rate wurde an den reduzierten Datensatz angepasst und vom eigentlichen Standardwert 0,001 auf 0,00001 reduziert. Als Loss-Funktion ist erneut die Categorical Crossentropy zum Einsatz gekommen. Mit einer Klassifizierungsperformance von 97% wurde das Arbeitspaket abgeschlossen.

BB.4 Bestimmung von Haar-und Hautfarbe Bei der Klassifizierung der **Haarfarbe** handelt es sich um ein Multiklassen-Problem, da mehr als zwei Haarfarben unterschieden werden. Die wichtigste Voraussetzung zur Umsetzung der Anforderung stellte auch dieses Mal wieder die Datengrundlage dar. Glücklicherweise konnte frühzeitig ein geeignetes Datenset identifiziert werden: Hierbei handelt es sich um das bereits zuvor erwähnte LFW-Datenset, welches auch genutzt wurde, um zu ermitteln, ob eine Person einen Bart trägt oder nicht. In Bezug auf die Haarfarbe stellt LFW Attributlabel für schwarze, blonde, braune und graue Haare sowie für das Merkmal einer Glatze zur Verfügung. Der Vorteil der Nutzung von LFW bestand darin, dass das Preprocessing zu großen Teilen mit denen des Bartproblems übereinstimmte, da die Bilddateien identisch waren und die Labelinformationen aus der gleichen Textdatei stammten. Bei der Verarbeitung der Attribute gab es

dennoch einige Abweichungen, die vor allem darauf zurückzuführen waren, dass es sich bei der Betrachtung der Haarfarbe nicht um ein binäres Problem handelt: Beim Preprocessing der Bart-Klassifizierung musste nur ein einziges Attribut betrachtet werden, bei der Haarfarbe waren es fünf verschiedene. Die Attributlabel in LFW wurden nicht händisch annotiert, sondern beruhen auf den Predictions eines Klassifizierungsmodells. Sie folgen somit keiner One-Hot-Kodierung, bestehen also nicht nur aus Einsen und Nullen, die An- bzw. Abwesenheit einer Merkmalsausprägung indizieren. Stattdessen wird für jedes Attribut ein Score angegeben, welcher die Distanz des Samples, also des Bildes, zu einem Stützvektor für die jeweilige Klasse angibt. Dabei ist es so, dass positive Werte die Anwesenheit und negative Werte die Abwesenheit der Attributausprägung implizieren. Für die One-Hot-Kodierung wurden dementsprechend alle fünf Attributwerte verglichen, der maximale wurde auf 1 alle anderen auf 0 gesetzt. Für den Fall, dass keine der Merkmalsausprägungen den Schwellwert von 0 überschreitet, wurde ein sechstes Label hinzugefügt. Dieses sollte alle anderen Farbklassen abdecken, für die keine Label vorlagen wie beispielsweise rote oder blaue Haare. Auf Basis der nun aufbereiteten Datengrundlage konnte ein Modell trainiert werden. Eine Besonderheit hierbei war, dass auf Klassengewichte zurückgegriffen wurde, um die nicht gleichmäßig verteilten Klassen auszubalancieren. Erneut wurden unterschiedlichste Architekturen getestet, die besten Ergebnisse lieferte ein auf Imagenet vortrainiertes ResNet50-Modell. Die Validation Accuracy nach dem Training betrug zirka 81%.

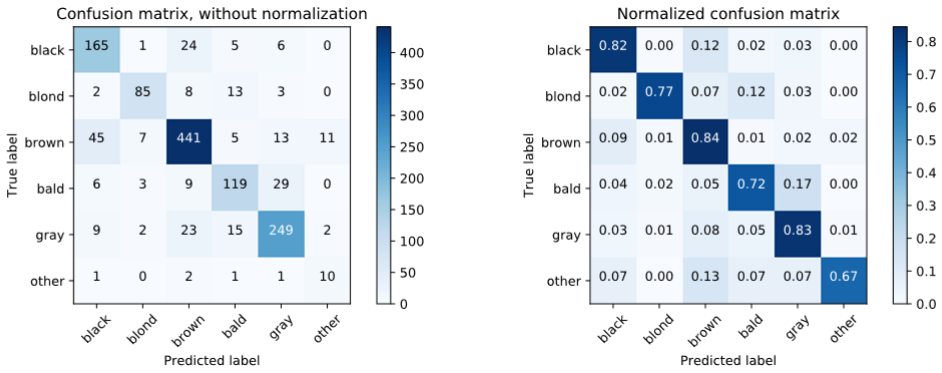


Abbildung 21: Klassifikationsmatrix Haarfarbe

Dies stellt ein durchaus zufriedenstellendes Ergebnis dar. Zum einen, da die Daten wie zuvor erwähnt nicht händisch annotiert wurden, zum anderen, da die Ermittlung der Haarfarbe allgemein kein triviales Problem ist aufgrund der Anfälligkeit von Farbklassifizierungsmodellen bei unterschiedlichen Belichtungsverhältnissen. Dies zeigte sich beim

praktischen Einsatz des Modells im Brillenberater und wird auch bei einem Blick auf die Klassifikationsmatrix in Abbildung 21 deutlich. So etwa treten gehäuft Fehlklassifizierungen zwischen schwarzen und braunen Haaren auf, was dadurch erklärt werden kann, dass etwa dunkelbraune Haare je nach Belichtung eher schwarz oder braun wirken können, was auch für das menschliche Auge bisweilen schwer zu unterscheiden ist. Schwierigkeiten treten auch auf bei blonden Haaren, die bei sehr hellem Licht teilweise nicht erkannt und fälschlicherweise als Glatze klassifiziert wurden, da sich der Ton der Haare kaum von der Haut unterscheidet. Weitere Probleme sind z. B. rote Haare („other“), die als braune klassifiziert werden und vice versa oder dunkelblonde Haare, die als braun erkannt werden. Da Haarfarben, die im Farbraum eng beieinander liegen und bisweilen schwer zu unterscheiden sind, im Brille24-Regelkatalog ohnehin von ähnlichen Regeln betroffen sind, wiegt der Großteil der Fehlklassifizierungen nicht sonderlich schwer. So würden z. B. einer Person mit schwarzen Haaren auch Brillen stehen, die einer Person mit braunen Haaren vorgeschlagen werden, wohingegen die Vorschläge für Personen mit blonden Haaren stärker abweichen. In Anbetracht der in diesem Kontext zufriedenstellenden Performance des Modells konnte das Arbeitspaket abgeschlossen werden.

Bei der Klassifizierung der **Hautfarbe** handelt es sich wie bei der Ermittlung der Haarfarbe um ein Multiklassen-Problem. Auch hier stellte sich das Auffinden eines geeigneten Datensets als Herausforderung dar. In Anbetracht fehlender Alternativen wurde sich dazu entschieden, auf das bereits bekannte UTKFace zu setzen. Dieses enthält streng genommen keine Informationen über die Hautfarbe, stattdessen sind verschiedene Ethnizitäten gelabelt. Da der Brille24-Regelkatalog jedoch an diesen Umstand angepasst wurde, stellte dies rückblickend kein Problem dar. UTKFace unterscheidet folgende Label:

- White
- Black
- Asian
- Indian
- Others (Hispanic, Latino, Middle Eastern)

Ein Vorteil der Nutzung von UTKFace bestand darin, dass dieses Datenset bereits zuvor zu Testzwecken für die Bestimmung von Alter und Geschlecht genutzt wurde und das Preprocessing somit übernommen werden konnte. Da größere Modelle keine nennenswerte Performancesteigerung hervorbrachten, kam als Architektur Mobilenet zum Einsatz,

wobei der Transfer-Learning-Ansatz mit Imagenet-Gewichten verfolgt wurde. Zudem wurden Klassengewichte verwendet, da die Klassen im Datenset nicht ausbalanciert waren. Die Performance des trainierten Modells auf dem Testset kann den Klassifikationsmatrizen aus Abbildung 22 entnommen werden.

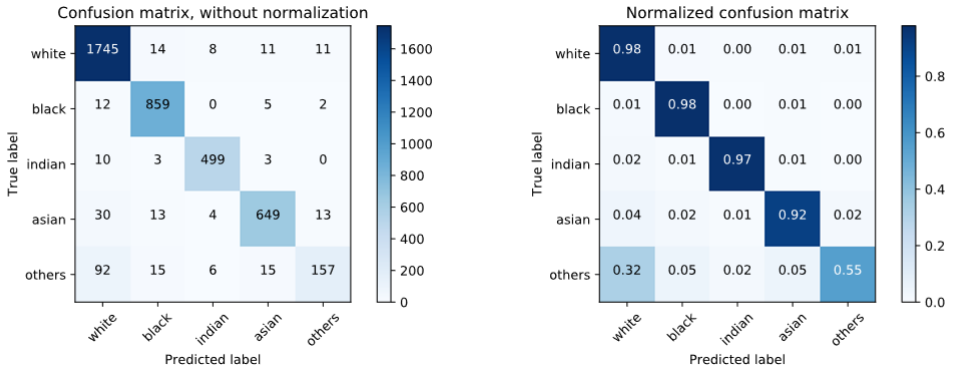


Abbildung 22: Klassifikationsmatrix Hautfarbe

Allgemein zeigt sich eine hohe Accuracy jenseits der 90% mit Ausnahme der Klasse „others“. Aus dieser Klasse werden 32% der Personen fälschlicherweise als „white“ klassifiziert. Dies kann damit erklärt werden, dass die Klasse „others“ im Datenset unterrepräsentiert ist und dies auch nicht durch die Verwendung von Klassengewichten komplett ausgeglichen kann. Im praktischen Einsatz des Modells im Brillenberater zeigte sich eine gewisse Anfälligkeit des Modells gegenüber unterschiedlichen Belichtungsverhältnissen. Trotz der Nutzung von Data Augmentation im Training mit besonderem Augenmerk auf der Änderung der Helligkeitswerte des betrachteten Bildes konnte diese Schwäche nicht komplett behoben werden. Sofern auf eine ausreichende Beleuchtung bei Aufnahme des Fotos geachtet wurde, konnten jedoch durchweg gute Ergebnisse erzielt werden.

Zwischenzeitlich wurde auch ein alternativer Ansatz zur Bestimmung verfolgt. Hierbei wurde versucht, die dominante Farbe im Bild durch Clustering der Pixel zu extrahieren. Dies war möglich, da die Bilder in UTKFace sehr eng gecroppt sind, dies bedeutet, dass in der Regel kein Hintergrund und keine Haare in dem Bild zu sehen sind, sondern nur das Gesicht, wobei die Haut einen großen Teil dieses Ausschnitts einnimmt. Auch bei diesem Ansatz zeigten sich jedoch deutliche Probleme mit den Belichtungsverhältnissen in den Bildern, was dazu führte, dass die extrahierten dominanten Hautfarben in der Regel Grautöne waren. Dieser alternative Ansatz wurde deswegen zugunsten der Klassifizierung mit UTKFace verworfen. Das Arbeitspaket wurde damit abgeschlossen.

BB.5 Bestimmung der Gesichtsform Die beiden folgenden Arbeitspakete beinhalten die Ergebnisse für die Bestimmung der Gesichtsform.

BB.5.1 Bestimmung von Facial Landmarks Da kein öffentlich verfügbares Datenset gefunden wurde, welches Label für alle 76 zuvor definierten Landmarks enthält, bedeutete dies, dass ein eigenes Datenset erstellt werden musste. Die Grundlage dieses Datensets stellten Bilder dar, welche von Brille24 zur Verfügung gestellt wurden. Ein jedes dieser Bilder war bereits mit 68 Landmarks versehen, das heißt, es mussten noch 8 Landmarks pro Bild nachträglich gelabelt werden. Dies wurde mithilfe des Tools „VGG Image Annotator“ (VIA) für die zirka 2000 Bilder zur Verfügung stehenden Bilder vorgenommen.

Auf Grundlage dieses Datensets sollte ein Modell trainiert werden, welches dazu dient, die 76 Landmarks vorherzusagen. Vor dem Training musste das Datenset allerdings zunächst bereinigt und das Preprocessing durchgeführt werden. Die zur Verfügung gestellten Bilder waren einzelne Frames, die von Videoclips entnommen wurden. Ihre Qualität schwankte stark, einige von ihnen waren bisweilen deutlich verpixelt, sodass die 8 zusätzlichen Landmarks nicht mit ausreichender Präzision gesetzt werden konnten und die Bilder beim Labeln übersprungen wurden. Zudem hatten die Bilder keine einheitliche Größe und waren nicht zugeschnitten. Als Konsequenz wurde zunächst die Face Detection auf die Bilder angewandt. Sofern auf einem Bild kein Gesicht erkannt werden konnte, wurde dieses übersprungen. Andernfalls wurde das Bild entsprechend der zurückgegebenen Bounding Box zugeschnitten. Da die Face Detection ein Gesicht normalerweise sehr eng croppt, sodass die Stirnregion in der Regel nicht erfasst wird, musste eine Margin hinzuaddiert werden, um einen größeren Ausschnitt des Bildes zu erhalten. Die Margin wurde auf 40% der Länge der Bounding Box gesetzt (es handelt sich um eine quadratische Bounding Box, Länge und Breite sind gleich). Diese Zahl orientiert sich an dem gecroppten IMDb-WIKI-Dataset, welches ebenfalls auf eine solche Margin zurückgreift. Da einige der Modelle (z. B. Geschlecht und Alter) mit eben diesem Datenset trainiert wurden, führt die Wahl zu einer gewissen Einheitlichkeit, da so für verschiedene Features auf das gleiche Preprocessing zurückgegriffen werden kann. Die Ermittlung der Landmarks wurde als klassisches Regressionsproblem behandelt. Jeder Facial Landmark ist ein Punkt in einem Bild, besitzt also eine x- und eine y-Koordinate. Bei einem Klassifikationsproblem ändert sich das Label nicht, wenn das Bild zugeschnitten wird. Die Landmarks hingegen müssen neu berechnet werden, da sich Größe und Ausschnitt des Bildes (und damit auch der Koordinatenursprung) ändern. Für einige der Bilder erwies sich eine 40% Margin als zu gering, sodass die berechneten Landmarks außerhalb des Bildes lagen. Dies trifft auf alle Land-

marks zu, die einen Wert haben, der kleiner oder gleich null bzw. größer als 224 ist (bei einer zugeschnittenen Bildgröße von 224x224). Auch diese Bilder wurden vernachlässigt, da dies gleichbedeutend damit ist, dass das durch den Landmark beschriebene Feature nicht im Bild zu sehen ist. Nach dem ersten Preprocessing waren noch zirka 1600 der ursprünglich gelabelten 2000 Bilder übrig. Diese stellten das finale Datenset für das Training eines ersten Modells dar. Als Architektur wurde zunächst MobileNet gewählt, wobei der Transfer-Learning-Ansatz verfolgt und die Imagenet-Gewichte geladen wurden. Der letzte Klassifikationslayer wurde entfernt, da dieser für die Klassifikation von Imagenet vorgesehen ist und eine Dimensionalität von 1000 Klassen bedient. Stattdessen wurden zwei Fully Connected bzw. Dense Layer mit 256 Neuronen angehängt, um die Kapazität des Netzes zu erhöhen. Als Outputlayer wurde dann ebenfalls ein Dense-Layer verwendet mit 152 Neuronen (76 Landmarks mal zwei Koordinaten pro Landmark). Die Aktivierungsfunktion dieses Layers ist linear, im Gegensatz zur Softmax-Aktivierung des ursprünglichen MobileNet-Modells, da es sich um eine Regression handelt. Kompiliert wurde das Modell mit Adam als Optimizer. Da die Landmarks in dem abgeschlossenen Intervall von 0 bis 1 normalisiert waren, wurde die Learning Rate zunächst mit 0,00001 relativ gering gewählt, wobei im Verlauf des Trainings mit verschiedenen Learning Rates experimentiert wurde. Der Regressionsproblematik bedingt wurden Mean Squared Error und der Mean Absolute Error als Loss bzw. Metrik auserkoren. Das erste Modell neigte aufgrund von fehlender Data Augmentation relativ schnell zu Overfitting und der denormalisierte mittlere absolute Fehler betrug etwa 5-6 Pixel bei einer Bildgröße von 224x224 Pixeln. Die grundlegende Form eines Gesichtes wurde zumeist korrekt erkannt, allerdings waren die Landmarks für präzise Berechnungen von Winkeln und Distanzmaßnahmen bisweilen deutlich zu ungenau platziert. Besonders davon betroffen waren Bilder, bei denen das Gesicht nicht frontal zur Kamera gerichtet war, sondern eine seitliche Neigung erkannt werden konnte.

Aufgrund dieser Ergebnisse wurde zunächst das Datenset bereinigt. Hierbei wurden alle Bilder aussortiert, bei denen die Person nicht frontal in die Kamera schaut, da hier bereits die ursprünglichen Landmarks von dlib nicht mit ausreichender Präzision gesetzt waren. Überdies wurde die Anzahl der betrachteten Landmarks zunächst auf 23, später auf 11 reduziert (Abbildung 23). Da Regionen wie Nase, Augen und Mund für die Bestimmung der Gesichtsform irrelevant sind, konnten die entsprechenden Punkte vernachlässigt werden.

Abbildung 23: Reduzierung der betrachteten Landmarks auf 23 bzw. 11

Zudem reduzierte dies die Dimensionalität des Output-Layers, was in der Theorie dazu führte, dass die übrigen Landmarks genauer predicted werden, da der Loss und somit die Anpassung der Parameter nur noch auf den für die Gesichtsform relevanten Landmarks beruhten. Die 23 dargestellten Landmarks stellen alle äußeren Landmarks dar, welche das Gesicht umrahmen. Die 11 Landmarks beruhen auf speziellen Punkten, die sich bei der Internetrecherche in zahlreichen Fashionblogs als wichtig herauskristallisiert haben und beschreiben Kinn, Kieferpartie, Stirn sowie Punkte an den Ohren und der Schläfe zur Bestimmung der Breite des Gesichts.

Das Problem der fehlenden Data Augmentation konnte dank der Library „imgaug“ behoben werden. Diese ist allgemein eine Library, welche viele Augmentationtechniken für Bilder im Bereich des maschinellen Sehens bereitstellt. In Bezug auf die hier beschriebene Anforderung ist imgaug allerdings auch in der Lage, mit Keypoints bzw. in diesem Fall im Speziellen Facial Landmarks umzugehen. Ein Beispiel: Wenn ein Bild, welches mit Landmarks versehen ist, im Rahmen der Augmentation rotiert wird, werden auch die Landmarks entsprechend rotiert. Ermöglicht wird dies aufgrund von affinen (Matrix-) Transformationen. Die Augmentation der Bilder mithilfe von imgaug wurde in der bereits beschriebenen „load batch“-Methode eingebaut. Hierbei wurde zudem sichergestellt, dass nur gültige Augmentations durchgeführt werden. Hiermit ist gemeint, dass Landmarks z. B. nicht aus dem Bild rausrotieren oder die Bilder zu weit in eine Richtung verschoben werden, sodass die Landmarks nicht mehr im erfassten 224x224 Pixel Bildbereich liegen. Dies wäre gleichbedeutend damit, dass das Feature, welches durch den Landmark beschrieben wird, nicht im Bild vorhanden ist, sodass dessen Position bestenfalls erahnt oder extrapoliert werden könnte. So können etwa die Landmarks an der Stirn nur schwer bestimmt werden, sofern die Stirn gar nicht mehr im Bild zu erkennen ist, da das Gesicht

im Rahmen der Data Augmentation nach oben verschoben wurde. Die Sicherstellung der Erzeugung gültiger Landmarks war relativ simplen Charakters: Nach der Augmentation wurde geprüft, ob sich alle erzeugten Landmarks im Intervall von 0 bis 224 befinden. Sofern dies nicht der Fall war, wurde das Ausgangsbild erneut augmentiert. Dies wurde in einer Schleife so lange wiederholt, bis eine gültige Augmentation entstand. Durch eine geeignete Wahl der Parameter für die einzelnen Techniken wie Rotation, Verschiebung etc. konnte sichergestellt werden, dass die Performance während des Trainings der Modelle aufgrund dieses Vorgangs nicht zu sehr beeinträchtigt und in der Regel nach spätestens zwei Schleifendurchläufen eine gültige Augmentation erzeugt wurde. Die Ergebnisse einer solchen Augmentation sind in der nachfolgenden Abbildung beispielhaft visualisiert.

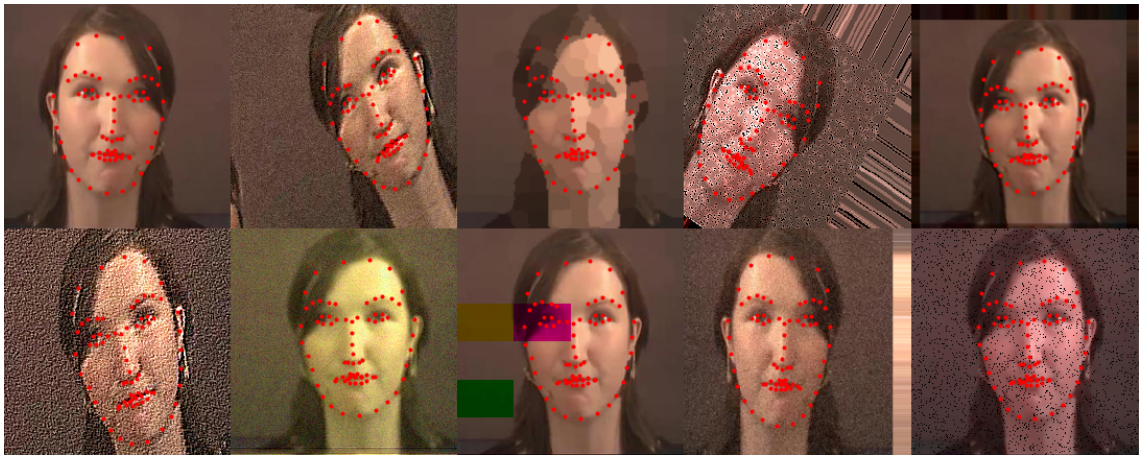


Abbildung 24: Beispielhafte Data Augmentation mit imgaug

In der linken oberen Ecke zu sehen ist das Originalbild, welches als Input in der „load batch“-Methode geladen und schließlich mithilfe von imgaug augmentiert wird. Die übrigen 9 Bilder sind Resultate der Augmentation, wobei diese aus einer Kombination unterschiedlichster Techniken entstanden sind. Einige Beispiele hierfür sind:

- Horizontale Spiegelung
- Rotation des Bildes
- Änderung der Helligkeit
- Änderung der Bildschärfe
- Ersetzen von Pixelclustern durch Superpixel

- Gaußsches Rauschen
- Cropping und Padden des Bildes
- Änderung der Farbsättigung
- Dropout, also das zufällige Auslassen von Pixeln

Selbstredend wurden nicht immer alle Techniken gleichzeitig angewendet, sondern es wurde jeweils eine Eintrittswahrscheinlichkeit definiert oder es wurde festgelegt, dass aus einer definierten Liste von Methoden eine gewisse Anzahl n zufällig ausgewählt werden soll. Die hier gezeigten Beispiele sind nur ein kleiner Ausschnitt für eines der Bilder aus dem Datenset. Tatsächlich wurden die Landmark-Modelle in der Regel für einige hundert Epochen trainiert, wobei in jeder Epoche ein einzigartiges Bild kreiert wird. Dadurch wird sichergestellt, dass das Modell die Position der Landmarks nicht auswendig lernen kann, denn wie auch in der Abbildung zu erkennen ist, wird die Position der Landmarks an die Transformationen angepasst. Dies war insgesamt betrachtet ein entscheidender Schritt, um dem Overfitting der Modelle entgegenzuwirken.

Mit den zuvor beschriebenen Anpassungen wurde dann neue Modelle trainiert, wobei sich Architektur und Parameter im Vergleich zum vorherigen Modell initial nicht änderten, im Verlauf des Trainings wurden dann verschiedene Änderungen vorgenommen, wie beispielsweise die Reduzierung der Learning Rate, um den Validation Loss so weit wie möglich zu verringern. Auch wurde im Verlauf der Zeit mit vielen anderen Architekturen als MobileNet experimentiert, wobei Densenet und Resnet bessere Ergebnisse als das ursprüngliche Modell erzielen konnten und aufgrund der größeren Kapazität der Netze zeigte sich eine schnellere Konvergenz beim Training. Das beste Modell am Ende des Trainingsvorgangs basierte letztendlich auf InceptionV3. Dieses stellte eine deutliche Verbesserung dar, da die Landmarks nun mit einem Mean Absolute Error von zirka 1,6 Pixeln predicted werden konnten. Bei einer Bildgröße von 224x224 Pixeln entspricht dies einem prozentualen Fehler von etwa 0,7%. In der Zwischenzeit erhielt die Projektgruppe vom Praxispartner Brille24 den versprochenen Regelkatalog zur Ermittlung der Gesichtsform anhand der Facial Landmarks. Folgende Gesichtsformen werden dabei unterschieden:

- diamond
- heart
- oblong

- oval
- round
- square

Für die Bestimmung der Gesichtsform werden zunächst verschiedene Distanzmaße und Winkel berechnet wie die Länge und Breite des Gesichts, die Breite der Stirn, die Breite der Kieferpartie oder der Winkel zwischen Kiefer und Kinn. Anhand verschiedener definierter Thresholds, die auf experimenteller Basis bestimmt wurden, erfolgt dann eine Zuordnung zu einer bestimmten Gesichtsform. Dies erwies sich jedoch als problematisch. Die Thresholds waren hart codiert und ließen wenig Spielraum für mögliche Fehler. Da die Landmarks trotz des besseren Modells für diesen Anwendungsfall immer noch nicht robust und präzise genug predicted wurden, schien die Bestimmung der Gesichtsform in der praktischen Anwendung des Brillenberaters eher dem Zufall überlassen zu sein. Bereits leichte Neigungen des Gesichtes oder andere perspektivische Änderungen führten zu deutlich unterschiedlichen Ergebnissen, sodass einer Person, die mehrmals hintereinander den Brillenberater testete, in den seltensten Fällen dieselbe Gesichtsform zugeordnet wurde. Auch bei Personen mit sehr markanten Gesichtsformen, dabei insbesondere einer ausgeprägte (Unter-) Kieferkontur, wies das Modell Schwächen auf. Die markanten Konturen wurden in der Regel nicht erfasst, da das Modell scheinbar implizit immer von einem ovalen Gesicht auszugehen schien.

Nach den Erkenntnissen der Zwischenpräsentation sollte das bestehende Modell zur Vorhersage der Landmarks, welches auf Basis des von Brille24 bereitgestellten Datensatzes trainiert wurde, zur Bewertung der Performance auf einem bisher fremden Datensatz evaluiert werden. Es ist ein Datensatz gesichtet worden, in welchem Annotationen für die Gesichtsform hinterlegt sind. Dieser besteht aus ca. 500 Bildern, jede der fünf Kategorien (Rund, Eckig, Oval, Lang, Herz) ist mit je 100 Bildern im jeweiligen Datensatz hinterlegt. Da es sich im Falle der je 100 Bildern pro Kategorie um Beispiele handeln, welche relativ disjunkt voneinander sind, sollte dieser Datensatz dafür verwendet werden, um das bestehende Landmark-Modell zu bewerten. Ein weiterer Ansatz, der mit diesem Vorhaben verknüpft war, war die Visualisierung prägnanter Unterschiede innerhalb der Gesichtsformen. Der Plan war es, das bestehende Landmark-Modell eine Vorhersage der Landmarks auf jedem der 500 Bilder durchführen zu lassen, die Koordinaten dann anschließend mithilfe eines Scatter-Plots zu visualisieren und schließlich Regionen zu identifizieren, in denen Unterschiede bei der Prognose der Koordinaten bestehen. Jedes der 100 Bilder je Kategorie wurde weiterhin in unterschiedlichen Farben geplottet, um die jeweilige Kategorie

ebenfalls voneinander abgrenzen zu können.

In der Abbildung 25 kann ein solcher Plot am Beispiel der herzförmigen Gesichter betrachtet werden. Jeder der Punkte repräsentiert die Vorhersage einer x- und y-Koordinate des Landmark-Modells.

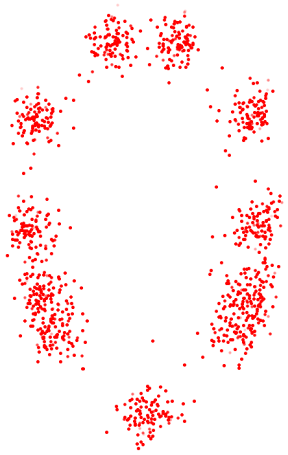


Abbildung 25: Landmark-Plot für herzförmige Gesichter

Anzumerken ist an dieser Stelle, dass es sich innerhalb des betrachteten Plots nicht um die vollständigen 100 Bilder mit jeweiliger Annotation handelt, sondern um ca. 30 händisch ausgewählte Exemplare, die einen Extremfall in ihrer Kategorie darstellen. Damit sollte ermöglicht werden, dass potenzielle Unterschiede nochmals deutlicher erkannt werden können. Diese Anmerkung greift auch für die restlichen Kategorien. Das weitere Vorgehen bestand nun darin, nach und nach die Plots der Prognosen des Modells zur jeweiligen Kategorie in einer anderen Farbe in den bestehenden Plot einzufügen. Das Ergebnis nach Hinzufügen der Vorhersage für Bilder mit ovalen Gesichtern sieht demnach wie folgt aus.

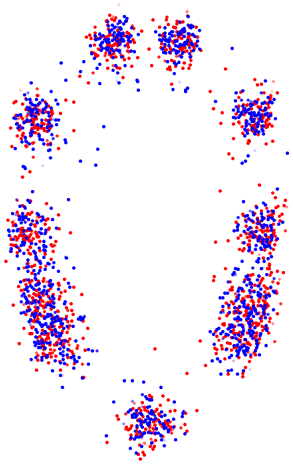


Abbildung 26: Landmark-Plot für herzförmige und ovale Gesichter

Zunächst einmal ist auffällig, dass nichts auffällig ist. Zwar liegen bei beiden Kategorien in den Prognosen einzelne Ausreißer vor, betrachtet man jedoch die Ballungsregionen der prognostizierten Koordinaten je Kategorie wird man feststellen, dass keine relevanten Unterschiede vorliegen. Dies könnte aber auch damit zusammenhängen, dass es einfach keine relevanten Unterschiede zwischen diesen beiden Kategorien gibt, was sich allerdings nach eigener Betrachtung der Bilder als falsch herausgestellt hat. Fügt man nun die Vorhersagen für die Bilder der übrigen Kategorien hinzu, ergibt sich folgendes Bild.

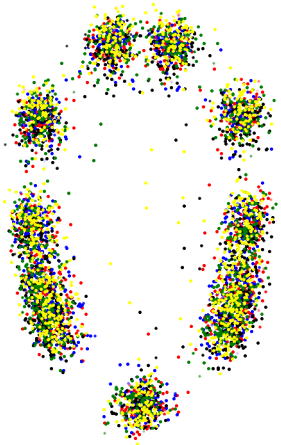


Abbildung 27: Landmark-Plot für alle Gesichtsformen

In dieser Abbildung kann nun die Vorhersage der Landmarks aller Bilder je Kategorie be-

trachtet werden. Unmittelbar wird deutlich, dass in den Plots der Vorhersagen des Modells je Kategorie keinerlei Unterschiede zu erkennen sind. Das Modell prognostiziert für jedes Bild, unabhängig der Kategorie, die gleichen Koordinaten. Dies ist ein immenses Problem, da die genaue Unterscheidung der Gesichtsform einer jeweiligen Person eins der relevantesten Filterungskriterien bei der Generierung von personenbezogenen Produktvorschlägen darstellt. Aus diesem Grund musste das bestehende Landmark-Modell überarbeitet werden, damit es effektiv zwischen Gesichtsformen unterscheiden kann.

Das Team hat zwei potenzielle Aspekte gefunden, welche der Ursprung für das vorhanden Problem sein könnten. Eine Annahme lag darin, dass die zum Trainieren des Netzes verwendete Datenquelle nicht ausreichend war. Nicht ausreichend bedeutet in diesem Falle, dass:

a) Zu wenig Daten zur Verfügung standen

b) Diese Daten nicht die entsprechende Qualität aufwiesen, um diesen komplexen Sachverhalt effektiv abzubilden

Die andere Annahme bestand darin, dass der Landmark-Ansatz als Zwischenschritt zur Bestimmung der Gesichtsform nicht funktioniert und andere Ansätze identifiziert und angewendet werden müssen.

Aus den zuvor genannten Annahmen des Problemursprungs konnten zwei Vorgehensoptionen abgeleitet werden. Das Team entschied sich dazu, beide parallel zu verfolgen, da die Bestimmung der Gesichtsform eines der wichtigsten Modelle für den Service des Brillenberaters darstellt. Eine Option bestand darin, ein neues Landmark-Modell zu trainieren und auf dessen Basis nochmals die Bilder mit den Gesichtsformen je Kategorie zu plotten, um mithilfe von Visualisierung relevante Unterschiede zu identifizieren. Die andere Option bestand darin, den Landmarkansatz aufzugeben und stattdessen das Problem mithilfe eines klassischen Klassifizierungsansatzes anzugehen.

Im Folgenden soll die Vorgehensoption beschreiben werden, welche das Trainieren eines neuen Modells umfasst. Hierfür wurden zwei unterschiedliche Ansätze verfolgt. Es wurde zum einen ein Modell auf Basis eines neuen Datensatzes trainiert. Zum anderen wurde ein komplett neuer Ansatz verfolgt, der die Prediction der Landmarks nicht als Regressionsproblematik behandelt: Der YOLO-Ansatz. Dieser wird nachfolgend genauer erläutert.

YOLO-Ansatz für die Facial Landmarks

Der YOLO-Ansatz ist ein weiteres Verfahren zur Bestimmung der Landmarks, um daraus im nächsten Schritt die korrekte Gesichtsform des Kunden abzuleiten. Der Grund für die Entwicklung eines weiteren Verfahrens sind die unzureichenden Ergebnisse im Bereich des Kinns und der Wange. Da die Landmarks in dieser Gesichtspartie von besonderer Bedeutung für die Klassifizierung der Gesichtsform sind, soll ein Modell entworfen werden, das in diesem Bereich noch geringere Abweichungen von den Ground Truth Werten zulässt, als der bisherige Ansatz.

Der Grundgedanke des hier entwickelten YOLO-Ansatzes besteht darin, dass die Vorhersage der Landmarks nicht mehr als Regressionsproblem, sondern als Problem der Object Detection formuliert wird. Bei dem bisherigen Regressionsmodell werden die Koordinaten der Landmarks im Fully-Connected-Layer des neuronalen Netzes ermittelt und dann ausgegeben. Dieses Vorgehen ermöglicht in der Regel eine hohe Generalisierbarkeit und robuste Ergebnisse, da Abhängigkeiten zwischen den einzelnen Landmarks in der Berechnung berücksichtigt werden können. Diese Korrelationen zwischen den Landmarks führen allerdings auch zu einer Vielzahl an leichten Abweichungen [WJ19].

Um die Einschränkungen des Fully Connected Layers zu umgehen, wird die YOLO Object Detection geringfügig modifiziert und dadurch für die Landmarks nutzbar gemacht. Der Grundgedanke des YOLO-Ansatzes besteht darin, das Eingangsbild in Raster einzuteilen und für jedes dieser Teile eine Vorhersage zu treffen, ob ein Objekt vorhanden ist und wo es innerhalb des Rasters liegt [Red+15]. Da für jedes Raster nur ein Objekt erkannt werden kann, wird die Gesamtzahl der Landmarks auf die 11 Landmarks reduziert, die für die Bestimmung der Gesichtsform von besonderer Bedeutung sind.

Architektur: Die Architektur des implementierten Netzes besteht aus einem vortrainierten VGG-Backbone, welches eine Eingangsgröße von 224*224*3 erwartet. Von dieser Grundlage werden alle Schichten nach dem vierten MaxPooling abgeschnitten und durch eine neue Ausgabeschicht ersetzt. Die letzte Schicht ist ein Convolutional Layer, welche aus 14 Filtern besteht, die jeweils einen Parameter besitzen. Die Filter setzen sich aus den folgenden Werten zusammen:

- 1 Prediction Score
- 2 Koordinaten

- 11 Parameter für die Klassifizierung der Punkte

Der Prediction Score ist binär kodiert und gibt an, ob sich in dem entsprechenden Raster ein Objekt befindet. Daneben existieren zwei Werte für die x- und y- Koordinaten. Diese liegen zwischen 0 und 1 und geben dadurch an, an welcher Stelle sich das Objekt innerhalb der Rasterzelle befindet. Die verbleibenden 11 Parameter werden für die Klassifizierung der Objekte verwendet, um im Postprocessing sicherzustellen, dass jeder der 11 Gesichtspunkte nur einmal zurückgegeben wird.

Trainingsdaten: Für das Training der Architektur stehen 824 Bilder mit der zugehörigen Annotation zur Verfügung. Im Rahmen des Preprocessings müssen die 11 Landmarks entsprechend umgerechnet werden, dass der Output zu der Architektur passt. Die Form der Ground Truth Werte für das Training ist 28*28*14 und dementsprechend ist für jede Rasterzelle eine Vorgabe mit 14 Parametern gegeben. In Abbildung 28 sind drei Beispiele für die Trainingsdaten abgebildet und mit den Grenzen der Rasterzellen versehen.

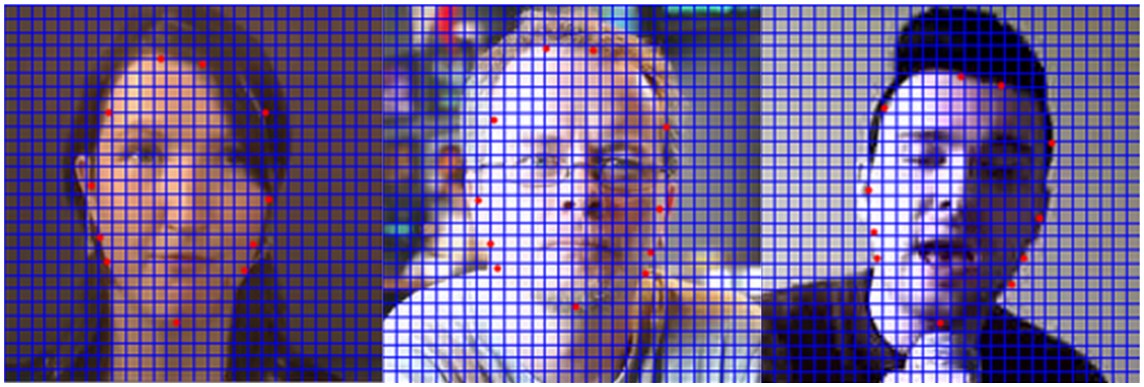


Abbildung 28: Trainingsdaten für den Yolo-Ansatz (Beispiel)

Training: Für den Trainingsprozess wird ein Generator entworfen, der die Bilder bereitstellt. Innerhalb dieser Routine werden die Daten auch vorverarbeitet. Zum einen werden Bilder und Landmarks augmentiert und zum anderen erfolgt das beschriebene Preprocessing, damit die Daten in der korrekten Form für das Training bereitstehen. Für das eigentliche Training werden jeweils vier Bilder innerhalb eines Batch eingelesen. Zu Beginn des Trainings werden die vortrainierten Gewichte des VGG-Teils eingefroren und die verbleibenden Schichten der Architektur können für 20 Epochen bei einer Learning Rate von 0,001 trainiert werden. Anschließend werden alle Ebenen für das Training freigegeben und die Learning Rate reduziert sich auf 0,0005. Dieses Vorgehen ermöglicht es, dass die

Informationen aus dem vortrainierten VGG-Teil optimal genutzt werden können. Nach insgesamt 150 Epochen stellt sich ein stabiler Zustand ein und der Loss stagniert bei 0,00012.

Loss: Während der Loss bei Klassifikations- oder Regressionsproblemen in der Regel nur als Parameter in der Trainingsmethode festgelegt wird, muss für diesen Ansatz eine eigene Loss-Funktion geschrieben werden, welche im folgenden Ausschnitt aufgeführt ist.

```
def yolo_loss(y_true, y_pred):
  lambda_obj = 10
  lambda_coord = 2
  lambda_noobj = 1
  lambda_class = 1

  obj_true = y_true[...,0]
  noobj_true = 1 - obj_true
  coord_true = y_true[..., 1:3]
  class_true = y_true[..., 3:-1]

  obj_pred = K.sigmoid(y_pred[..., 0])
  noobj_pred = 1 - obj_pred
  coord_pred = y_pred[..., 1:3]
  class_pred = K.softmax(y_pred[..., 3:-1])

  loss1 = (lambda_obj * K.binary_crossentropy(obj_true, obj_pred)) / (28*28)
  loss2 = (lambda_noobj * K.binary_crossentropy(noobj_true, noobj_pred)) /
    (28*28)
  loss3 = lambda_coord * obj_true * K.mean(K.square(coord_true - coord_pred)
    , axis=-1)
  loss4 = lambda_class * obj_true * K.categorical_crossentropy(class_true,
    class_pred)

  loss = K.mean(loss1 + loss2 + loss3 + loss4) / 16
return loss
```

Listing 11: Loss-Funktion des YOLO-Ansatzes

Der Loss setzt sich aus folgenden Bestandteilen zusammen:

- Loss1: Der erste Teil gibt den Fehler des Prediction Scores an, wenn sich ein Objekt in dem Raster befindet. Der Fehler wird durch die Binary Crossentropy angegeben.
- Loss2: Analog zum ersten Teil-Loss wird der Prediction Score beurteilt. Allerdings bezieht sich der Teil auf die Rasterzellen, in denen sich kein Objekt befindet.

- Loss3: In diesem Teil wir die Genauigkeit der Koordinaten anhand des Mean squared errors bemessen.
- Loss4. Der letzte Abschnitt der Lossfunktion betrifft die Klassifizierung der Landmarks und verwendet die Categorical Crossentropy als Fehlermaß.

Alle vier Abschnitte der Lossfunktion werden zusätzlich durch Gewichte bewertet, welche dem klassischen Yolo-Ansatz entnommen werden.

Die Ergebnisse dieses Ansatzes zeigen, dass die Vorhersage von Landmarks grundsätzlich nicht nur durch Regressionen lösbar ist. Allerdings existieren auch Einschränkungen, weshalb dieses Verfahren in der Folge nicht weiterverfolgt wird. Die Qualität der Ergebnisse ist insbesondere in der Stirnregion nicht ausreichend und es besteht die Möglichkeit, dass nicht alle 11 Landmarks vorhergesagt werden können.

Regressionsansatz auf Basis eines neuen Datensets

Für diesen Ansatz wurde das Youtube-Faces-Datenset verwendet, in welchem Annotationen für jeweilige Landmarks der entsprechenden Person hinterlegt sind. Das Datenset besteht aus einer Vielzahl von Videos, in welchen jeder Frame mit entsprechenden Annotationen hinterlegt ist. Ein Beispiel kann in der folgenden Abbildung 29 betrachtet werden.

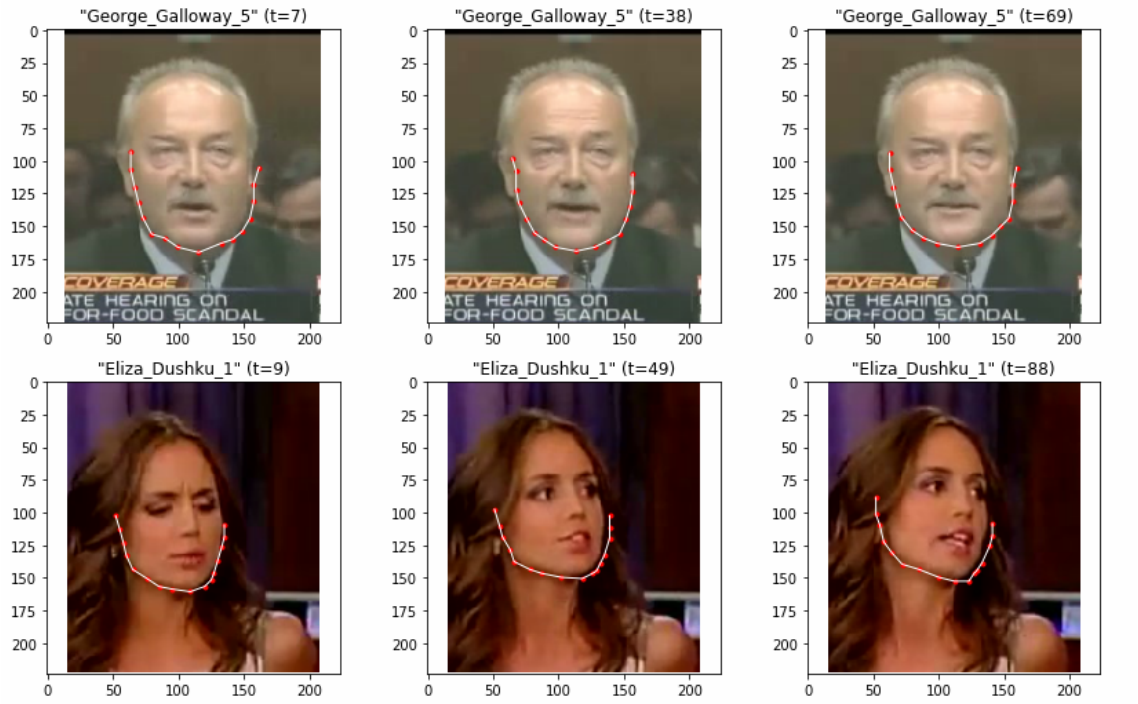


Abbildung 29: Beispielhafte Frames aus dem Youtube-Faces-Datenset mit zugehörigen Annotationen

Im Bild zu sehen sind sowohl die annotierten Landmark als auch die Namen und ein Zeitstempel, welcher den Zeitpunkt des Frames innerhalb der Aufnahme beschreibt. Es ist anzumerken, dass in dem Datenset wesentlich mehr Annotationen für Landmarks vorhanden sind, wie bspw. Landmarks für die Augen- und Nasenregion. Allerdings sind diese Landmarks unrelevant für den von uns betrachteten Anwendungsfall, weshalb nur die Landmarks der äußeren Gesichtsregionen zum Trainieren des Netzes verwendet werden.

Anders als in bisherig verwendeten Datensets, in welchen die Bilder als separate Dateien oder in Form einer Pickle-Datei abgelegt wurden und entsprechende Annotationen in einer bspw. beigelegten CSV-Datei hinterlegt waren, unterschied sich die Struktur innerhalb des Youtube-Faces-Datensatzes enorm. Hier waren die Daten in extrem verschachtelten .NPZ-Files hinterlegt, weshalb wesentliche Änderungen in Hilfsmethoden, wie dem batchweise Laden der Dateien, vorgenommen werden mussten. Das Training des Models gestaltete sich jedoch ähnlich zum Training vorheriger Modelle. Als Architektur wurde InceptionV3 mit vortrainierten Gewichten verwendet. Die zuvor abgeschnittenen letzten beiden Layer wurden durch zwei Dense-Layer mit je 256 Units und einer ReLu-Aktivierung ersetzt.

Das Anfügen eines zusätzlichen MaxPooling-Layers sollte zudem potenzielles Overfitting einschränken. Als Output-Layer wurde ein Dense-Layer mit je 32 Input-Units verwendet, da lediglich 16 Punkte vorhergesagt werden sollten und diese Punkte aus je x- und y-Koordinate bestehen. Da es sich um ein Regressionsproblem handelt, wurde eine einfache lineare Aktivierung im letzten Layer verwendet. Um die relative Abweichung der Prognose vom tatsächlichen Wert der Koordinate zu erhalten, wurde der Mean-Absolute-Error als Metrik eingesetzt, um die Performance des Netzes auch während des Trainingsprozesses nachvollziehen zu können. Die besten Ergebnisse wurden zudem unter Verwendung des ADAM-Optimizers mit der Standard-Learning-Rate von 0,001 in Kombination mit dem Mean-Squared-Error als Loss-Funktion erzielt. Mit einer durchschnittlichen Abweichung von ca. 2 Pixeln bei der Bestimmung der jeweiligen Landmarks wurde das Arbeitspaket abgeschlossen. Trotz des vermeintlich leicht höheren Mean Absolute Erros von 2 Pixeln im Vergleich zum besten Modell, welches mit dem Brille24.Datensatz trainiert wurde (1,6 Pixel), zeigten sich in der praktischen Anwendung deutlich bessere Resultate. Die Landmarks schienen deutlich robuster predicted und dabei z. B. auch von kleineren Abweichungen der Kopfneigung nicht beeinträchtigt zu werden. Zudem ist das Modell in der Lage, bei Personen mit einer markanten Gesichtsform deutlich präzisere Landmarks zu predicten und die Konturen zu erfassen. Ein Beispiel eines Härtefalls, bei welchem das alte Modell noch mangelhafte Ergebnisse lieferte, ist in der folgenden Grafik 30 zu sehen, die Predictions des neuen Modells sind dazu im Vergleich ebenfalls dargestellt.

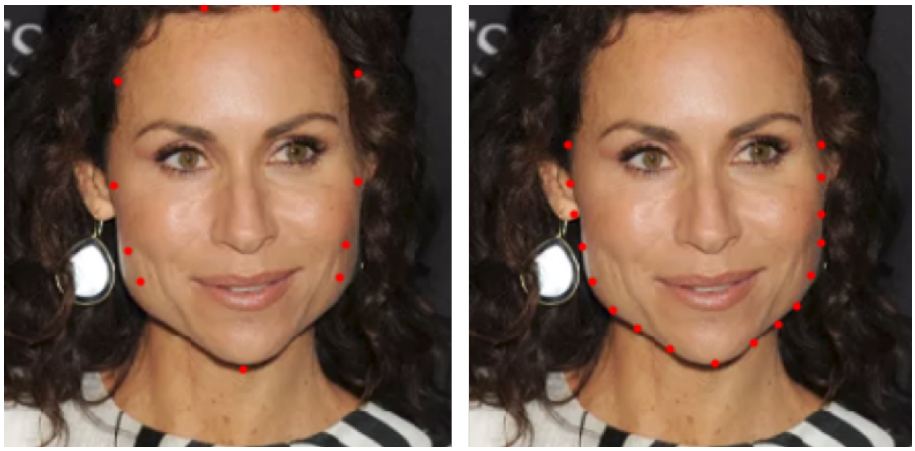


Abbildung 30: Vergleich der Landmark-Modelle

Wie zu erkennen ist, wird die markante Kieferparte vom neuen Modell deutlich besser erfasst, was für die Errechnung von Winkeln und Distanzen mithilfe der Landmarks von

eminenter Wichtigkeit ist. Für die Stirnpartie hingegen fehlen zwecks nicht vorhandener Label die Landmarks. Dies stellt jedoch kein Problem dar, da diese Landmarks vom alten Modell in der Regel zuverlässig vorhergesagt werden konnten und die Ergebnisse der beiden Modelle somit im Zweifelsfall kombiniert werden können, sofern die Aufgabe dies erfordert.

Nachdem nun ein zuverlässiges Modell für die Prediction der Landmarks trainiert wurde, konnte mit dem eigentlichen Vorhaben der Evaluierung und der Visualisierung der prognostizierten Landmarks je Kategorie auf dem bereits zuvor verwendeten Datenset begonnen werden. In der Abbildungen 31 können die vorhergesagten Landmarks des neuen Modells betrachtet werden. Von links nach rechts betrachtet kann zum einen ein Beispiel für ein eckiges Gesicht, ein ovales Gesicht sowie ein rundes Gesicht veranschaulicht werden.

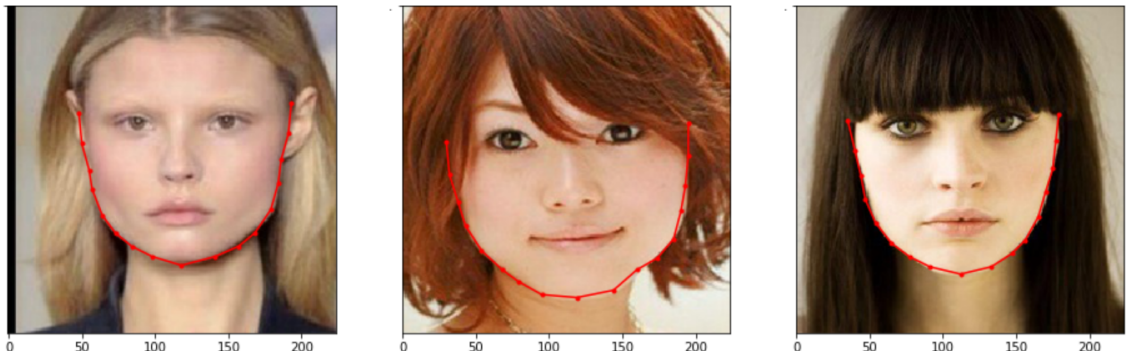


Abbildung 31: Vorhergesagte Landmarks für unterschiedliche Gesichtsformen

Bereits mit dem bloßen Auge lassen sich bezüglich einiger Ausprägungen der jeweiligen Gesichtsform deutliche Unterschiede erkennen. Bei eckigen Gesichtern ist beispielsweise die Entfernung der Landmarkkoordinaten der Wangenknochen am größten, während bei runden Gesichtern die Entfernung zwischen den Landmarkkoordinaten der Ohrenbereiche am größten ist. Diese Unterschiede sollen nun jedoch neben einer visuellen Repräsentation auch quantitativ messbar gemacht werden, indem ausgewählte Berechnungen mithilfe der ermittelten Koordinaten durchgeführt werden. Die Grundlagen dieser ausgewählten Berechnungen können in den Beispielen aus Abbildung 32 betrachtet werden.

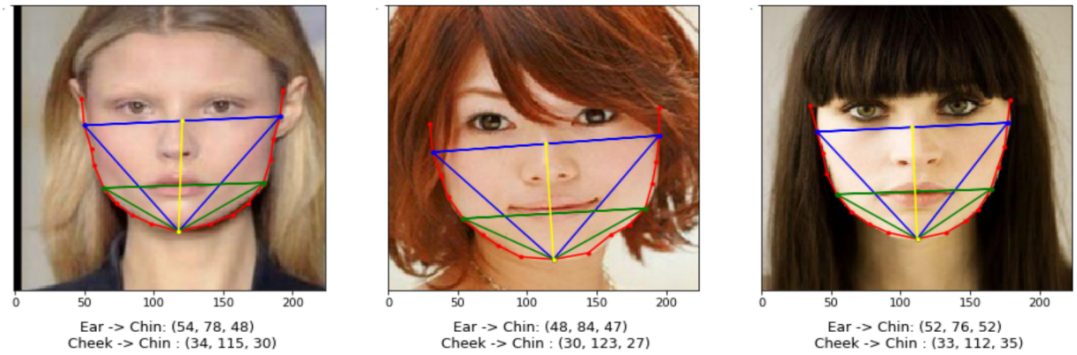


Abbildung 32: Berechnungsgrundlagen

Der Fokus wurde vor allem auf die Kinn-, Kiefer- sowie Ohrenregion gelegt, da dort bereits durch reine Betrachtung die größten Unterschiede innerhalb der Kategorien identifiziert werden konnte. Neben den reinen Abständen der beiden Ohren- sowie Kieferpunkte wurden ebenfalls die Abstände der Kinnpunkte zu Kiefer-bzw. Ohrenpunkte berechnet. Bereits hier konnte festgestellt werden, dass sowohl der Abstand zwischen den beiden Ohrenpunkten als auch der Abstand der beiden Kieferpunkte bei den runden Gesichtern am größten ist. Bei den eckigen Gesichtern war es vor allem der Abstand der beiden Kieferpunkte, welcher das eckige Gesicht vom Ovalen unterscheidbar machte. Die gelbe Diagonale beschreibt den Abstand des Kinnpunktes zur Ohrenregion. Hier weist im Schnitt das ovale Gesicht den größten Abstand auf, wobei das runde Gesicht den minimalsten Abstand besitzt. Diese Informationen spiegeln sich auch in den berechneten Winkeln der in den Beispielbildern gezeigten Dreiecken wider. Mithilfe dieser Informationen sollten nun Wertebereiche definiert werden, die genau die zuvor genannten Sachverhalte abdecken, um anschließend auf Basis dieser Grenzwerte ein jeweiliges Gesicht einer Kategorie zuzuordnen. Allerdings ist eine solche Definition von Wertebereichen ohne einheitliche Skalierung unmöglich. Der Fotoupload des Brillenberaters gibt dem Kunden keinerlei Vorgaben, aus welchem Winkel, aus welcher Position oder aus welchem Abstand zur Kamera das hochzuladende Foto gemacht werden muss. Im Kontext der Definition von Wertebereichen der jeweiligen Gesichtsformkategorien ist dies ein Problem, da ohne einheitliche Skalierung keine allgemeingültigen Gegebenheiten abgebildet werden können und somit die Definition von Wertebereichen unmöglich ist.

Ein potenzieller Lösungsansatz für das Skalierungsproblem könnte bspw. eine Art Maske als Overlay innerhalb des Fotoaufnahmeprozesses sein, welches dem Nutzer vorgibt, sein

Gesicht innerhalb der Maske zu platzieren, um somit die gleiche Ausgangslage für jedes aufgenommene Bild zu ermöglichen. Um diesen Ansatz umzusetzen, fehlte allerdings letztendlich die Zeit, weshalb der verfolge Ansatz zur Bestimmung der Gesichtsform auf Basis von Landmarks beendet und durch einen klassischen Klassifizierungsansatz ersetzt wurde.

BB.5.2 Klassifizierung der Gesichtsform Wie bereits zuvor angeklungen ist die Bestimmung der Gesichtsform kein triviales Problem. Über einen langen Zeitraum hinweg konnte kein Datenset ausgemacht werden, welches gelabelte Gesichtsformen bereitstellt. Eine alternative Vorgehensweise bei Nichtvorhandensein von gelabelten Daten in ausreichender Qualität und Quantität ist es, ein eigenes Datenset aufzubauen. Bereits genutzte Datensets wie UTKFace, IMDb-Wiki oder auch APPA-REAL stellen Gesichtsbilder zur Verfügung, welche in der Theorie händisch mit der Gesichtsform annotiert werden könnten. Das Problem hierbei ist jedoch, dass die Gesichtsform oftmals nicht eindeutig ist und häufig auch der subjektiven Wahrnehmung der labelnden Person unterliegt. Dies zeigte sich, als der Vorschlag, ein eigenes Datenset aufzubauen, zur Debatte stand. Bereits bei den Testbildern, Portraitfotos der einzelnen Projektgruppenmitglieder, gab es oftmals Unstimmigkeiten bezüglich der Gesichtsformen und es konnte häufig kein Konsens gefunden werden, welches Label das vermeintlich korrekte ist. Es war dementsprechend auch nicht möglich, ein eigenes Datenset mit konsistent gelabelten Bildern aufzubauen.

Die Suche nach bereits gelabelten Datensets erwies sich in dem letzten Projektabschnitt letztendlich doch noch als erfolgreich. Im Rahmen ihrer Arbeit „A Hybrid Approach to Building Face Shape Classifier for Hairstyle Recommender System“ veröffentlichten Pasupa, Sunhem und Loo ein Datenset, welches jeweils 100 Bilder für die 5 Klassen

- heart,
- oblong,
- oval,
- round und
- square

beinhaltet [PSL19]. Mit Ausnahme des Labels „diamond“ finden sich also alle Gesichtsformen aus dem Brille24-Regelkatalog wieder. Um sicherzustellen, dass die Daten korrekt und konsistent gelabelt werden, gingen die Autoren folgendermaßen vor: Zunächst wurden 8 Freiwillige akquiriert, die sich bereitgestellt haben, um 1000 gecrawlte Bilder zu labeln.

Sie wurden zunächst gebeten, Expertenrichtlinien zur Bestimmung der Gesichtsform zu studieren. Anschließend wurden sie mit 10 Fotos (Fotos von zwei unterschiedlichen Personen pro Gesichtsform) konfrontiert, bei welchen ein Celebrity-Hairstylist die Gesichtsform bestimmte. Nur wenn die Probanden mindestens 8 der 10 Gesichtsformen richtig einschätzten, bestanden sie diesen initialen Qualifikationstest und durften die 1000 Bilder labeln. Dies traf auf 6 der 8 ursprünglichen Kandidaten zu. Von den 1000 Bildern wiederum wurden nur diejenigen für das finale Datenset ausgewählt, bei denen in mindestens 4 von 6 Fällen von den Label-Experten die gleiche Gesichtsform gewählt wurde. Dies traf auf 500 Bilder zu [PSL19].

Ein Nachteil am Datenset, neben der verhältnismäßig geringen Anzahl an Bildern, besteht darin, dass nur weibliche Personen betrachtet werden. Dies lässt sich damit erklären, dass die Autoren ein Recommender System für Frauenfrisuren aufbauen wollten und es somit keine männlichen Probanden benötigt. Beim Brillenberater ist dies anders, deswegen stellte sich die Frage, ob ein Modell, welches bei Frauen die Gesichtsform bestimmt, die nötige Transferfähigkeit besitzt und auch für Männer hinreichend gute Ergebnisse erzielt oder ob die fehlende Datengrundlage dies verhindert. Aufgrund der geringen Datenmenge wurde sich dafür entschieden, MobileNet als Architektur zu verwenden, um einem schnellen Overfitting entgegenzuwirken. Trotz dieser Maßnahme und trotz der Verwendung von Data Augmentation neigte das Modell sehr schnell zum Overfitting, da die Datenmenge schlichtweg zu gering war. Die besten Modelle erreichten knapp über 60% Accuracy auf dem Testset. Predictions auf vollständig neuen Bildern wie den bereits erwähnten Portraitfotos der Projektgruppen-Mitglieder offenbarten, dass das Modell nicht in der Lage war, zu abstrahieren, dabei war es egal, ob die Bilder Männer oder Frauen abbildeten.

Trotz des nicht zufriedenstellenden Klassifizierungsmodells erwies sich das Datenset als nützlich, da es von Experten gelabelte Ground-Truth-Label für die Gesichtsform bereitstellte. Dadurch konnten neue Ansätze verfolgt werden. Einer dieser Ansätze bestand in der bereits beschriebenen Verbesserung der Landmark-Predictions in Assoziation mit der Definition eigener Regeln zur Ableitung der Gesichtsform. Der andere Ansatz verstand sich als Versuch, die vorhergesagten Landmarks für eine Klassifizierung zu nutzen. Die Intuition dahinter lässt sich folgendermaßen erklären: Das zuvor erwähnte einfache Klassifizierungsmodell neigte wie erwähnt sehr schnell zum Overfitting, was an einer zu geringen Datenmenge auf der einen und einem zu großen Modell auf der anderen Seite liegt. Das Modell ist nicht dazu in der Lage, die Gesichtsform zu abstrahieren, sondern lernt gewissermaßen die Label zu den Gesichtsbildern auswendig. Die Gesichtsbilder selbst enthalten

viele Informationen respektive Features, die für die Bestimmung einer Gesichtsform nicht weiter relevant sein sollten, wie etwa Nase, Augen oder auch den Bildhintergrund. Die Idee bestand nun darin, die Landmarks mit dem überarbeiteten Modell vorherzusagen und diese als Input für ein weiteres Modell zu nutzen, welches genutzt wird, um die Gesichtsform zu klassifizieren. Auf diese Weise betrachtet das Modell nur die für die Gesichtsform relevanten Features aus dem Bild, welche durch die Landmarks beschrieben werden. Das Mapping der Landmarks zu einer Gesichtsform wird dann nicht explizit vom Menschen vorgenommen, sondern dem künstlichen neuronalen Netz überlassen, in der Hoffnung, dass dieses besser darin ist, bestimmte Abhängigkeiten, Gesetzmäßigkeiten oder Regeln zu erkennen, die für den Menschen nicht offensichtlich sind. Als Datenset wurden die 500 gelabelten Bilder von Pasupa, Sunhem und Loo verwendet. Für jedes dieser Bilder wurden mit dem verbesserten Landmark-Modell Predictions durchgeführt. Da dieses Modell auf dem Youtube-Datenset basiert, wird die Stirnpartie nicht berücksichtigt. Deswegen wird eine zweite Prediction durchgeführt mit dem alten Landmark-Modell. Die Outputs der beiden Modelle werden anschließend kombiniert und als Input für das nächste Modell verwendet. Zu jedem der Bilder, für welches die Landmarks vorhergesagt werden, liegen die Ground-Truth-Label für die Gesichtsform vor. Es liegen nun also 500 Listen von Landmarks mit einem dazugehörigen Gesichtsform-Label vor.

Zur Vorhersage benötigt es nun ein Modell, wobei sich in diesem Fall kein Convolutional Neural Network eignet, da der Input kein Bild, sondern eine einfache Liste von x- und y-Koordinaten, sprich Zahlen, ist. Somit wurde auch nicht auf ein vortrainiertes Modell wie MobileNet zurückgegeriffen, sondern die Architektur wurde selbst definiert. Als Gerüst wurde ein Multilayer Perzeptron genutzt, es wurden also mehrere Fully-Connected-Layer verbunden, wobei mit vielen verschiedenen Hyperparametern wie der Anzahl der Hidden Layer oder der Anzahl der Neuronen pro Layer experimentiert wurde. Zudem wurden Techniken wie Dropout und Regularisierung genutzt, um Overfitting entgegenzuwirken. Data Augmentation wurde im Vergleich zu den anderen Modellen nicht durchgeführt, da der Input aus Listen von Landmarks und nicht aus Bildern bestand. Retrospektiv betrachtet hätte das Datenset mit Hilfe der „imgaug“-Library jedoch künstlich vergrößert werden können, indem die Landmarks von augmentierten Bildern als weitere Input-Werte aufgenommen werden. Das finale trainierte Modell kam nicht über eine Accuracy von 50% hinaus. Es lässt sich nun nur spekulieren, ob die geringe Datenmenge oder etwa die Netzarchitektur ursächlich waren oder ob der Ansatz allgemein zum Scheitern verurteilt war.

In der Zwischenzeit wurde für Umsetzung anderer Anforderungen, nämlich die Klassi-

fizierungen von Bart/Nicht-Bart sowie der Haarfarbe, auf das Datenset „Labeled Faces in the Wild“ zurückgegriffen. Dieses stellt auch drei Label für die Gesichtsform bereit mit „oval“, „round“ und „square“. Die Label stellen allerdings selbst nur Predictions eines Modells dar und sind somit keine Ground-Truth-Label, die händisch annotiert wurden. Nichtsdestotrotz wurde auch mit diesem Datenset ein Modell trainiert zur Klassifizierung der Gesichtsform, um nachzuvollziehen, ob die größere Datenmenge eventuell zu einem Erfolg führt. Im Rahmen des Preprocessings musste zunächst eine One-Hot-Kodierung für die Label durchgeführt werden. Dafür wurden aus einer Textdatei, welche die Informationen zu 73 unterschiedlichen Attributen für jedes der Bilder enthält, zunächst die drei relevanten Attribute oval, round und square extrahiert. Die Prediction-Werte wurde dann verglichen. Der Gesichtsform mit der höchsten Prediction wurde der Wert 1 zugeordnet, die beiden anderen hingegen auf 0 gesetzt. In Bezug auf die Netzarchitektur wurde, wie so häufig, auf MobileNet gesetzt, auch die sonstige Vorgehensweise entsprach dem beschriebenen Standardvorgehen inklusive Data Augmentation, „load batch“-Methode etc.. Beim Training des Modells traten wieder relativ frühzeitig Probleme mit Overfitting auf, wobei die bekannten Techniken wie Anpassung der Learning Rate, Dropout oder mehr Data Augmentation keine Abhilfe schaffen konnten. Die Accuracy auf dem Testset lag bei zirka 60%, wobei bedacht werden muss, dass nur drei Klassen zur Auswahl standen, was die Performance des Modells sogar in einem noch schlechteren Licht erscheinen lässt. Dafür ursächlich scheint die schlechte Datenqualität zu sein: Nicht alle Gesichtsformen wurden im Datenset durch Label repräsentiert und es wurde immer der höchste Wert als Kriterium für das Label genommen, was zu Inkonsistenzen führen kann, wenn alle Predictions verhältnismäßig gering sind und das Gesicht offensichtlich weder rund, eckig noch oval ist. Zudem basieren die Label auf Predicitons, wodurch das trainierte Modell in Bezug auf die Performance ohnehin limitiert ist, da es maximal an die Leistung des Modells anknüpfen kann, welches die Attributlabel in „Labeled Faces in the Wild“ predicted hat. Zusammenfassend kann also auch dieser Ansatz, ein Klassifikationsmodell mit LFW zu trainieren, als gescheitert betrachtet werden.

Die finale Lösung des Problems der Bestimmung der Gesichtsform bestand in der Nutzung eines geeigneten Datensets. Der Praxispartner Brille24 befasste sich im Rahmen der Arbeit an einem internen Projekt ebenfalls mit der Extraktion der Gesichtsform aus einem Foto. Dabei stießen sie ähnlich zu den hier dargelegten Bemühungen auch gewissermaßen an ihre Grenzen, da Landmark-Ansätze nicht funktionierten und öffentlich verfügbare Datensets mit einer hinreichenden Quantität und Qualität an gelabelten Bildern nicht existieren. Aus diesem Grund wurde sich dazu entschieden, ein eigenes Datenset aufzubauen, wo-

bei bei Brille24 die nötige Fashion-Expertise vorhanden ist, um sicherzustellen, dass die Gesichtsform korrekt und konsistent gelabelt wird. Das Datenset selbst besteht aus zirka 15000 Bildern von Personen, wobei zwischen den Gesichtsformen heart, oblong, oval, round und square unterschieden wird. Besagtes Datenset wurde der Projektgruppe freundlicherweise zur Verfügung gestellt. Mithilfe dieser Datengrundlage konnten neue Modelle trainiert werden. Hierfür wurde sich wie gewohnt des Transfer Learning Ansatzes bedient. Als Architekturen kamen unter anderem MobileNet und InceptionV3 zum Einsatz, wobei ein Inception-Modell letztendlich die besten Resultate lieferte: Auf dem Testset konnte eine Accuracy von zirka 85% erzielt werden, was einer deutlichen Steigerung im Vergleich zu den zuvor beschriebenen Modellen gleichkommt. Details zur Performance des Modells können der Klassifikationsmatrix (Abbildung 33) entnommen werden.

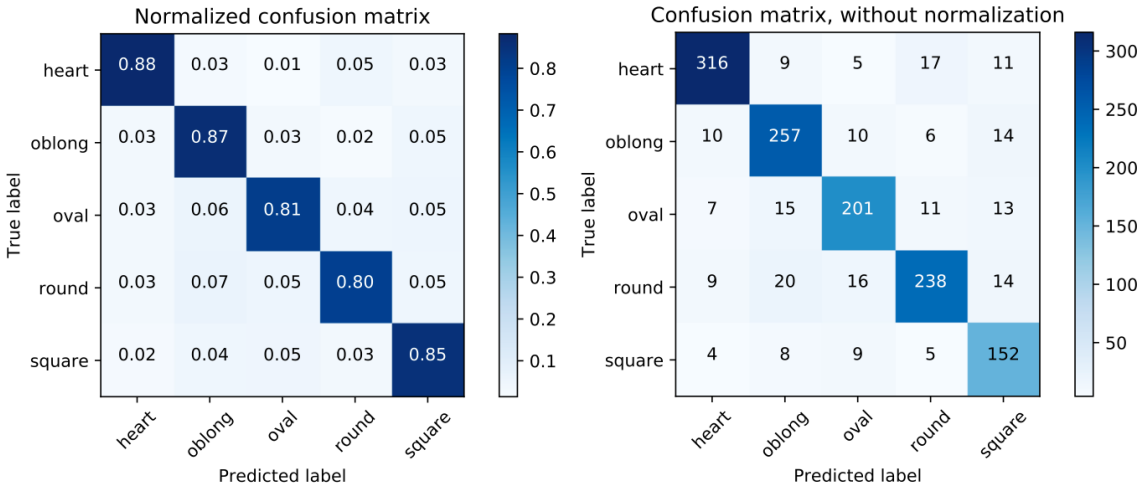


Abbildung 33: Klassifikationsmatrix für das Modell der Gesichtsform

Dieses erfreuliche Ergebnis stellte eine erneute Bestätigung dar, dass ein Machine-Learning-Modell mit der Datengrundlage steht und fällt. Auch im praktischen Einsatz innerhalb des Brillenberaters konnte sich das Inception-Modell zur Bestimmung der Gesichtsform bewähren. Im Rahmen einer Hausmesse bei der Volksbank konnten insbesondere im Vergleich zum vorherigen Stand, der Verwendung des Landmark-Ansatzes unter Anwendung der Brille24-Regeln zum Mapping zu einer Gesichtsform, deutlich bessere Resultate erzielt werden: Die extrahierte Gesichtsform entsprach in der Regel an subjektiven Eindrücken gemessen der Realität, was eine Bestätigung der Performance auf dem Testset bedeutet. Zudem wurde, sofern eine Person mehrmals den Berater testete, besagter Person die gleiche Gesichtsform zugeordnet. Der zufällige Charakter, welcher sich noch bei der dritten

Zwischenpräsentation offenbarte, war damit überwunden. Zusammenfassend konnte also nach einer langen, nicht unbeschwerlichen Reise die Anforderung zur Klassifizierung der Gesichtsform und folglich auch die übergeordnete Anforderung der Bestimmung der Gesichtsform (unabhängig des gewählten Ansatzes) erfüllt werden.

9.6.2 Beratung

Um eine personalisierte und individuelle Brillenempfehlung geben zu können, spielen unterschiedliche Faktoren eine Rolle. Im stationären Handel erfolgt dies durch die Erfahrung und Intuition des Optikers. Für die Überführung dieser Intuition und Erfahrung in die Online-Beratung sind gewisse Regeln erforderlich, welche auf ein Brillensortiment anwendbar sein müssen. Dafür war es notwendig, sich innerhalb der Projektgruppe intensiv mit den unterschiedlichsten Brillen auseinanderzusetzen und deren Attribute und Eigenschaften genau zu beschreiben.

Brillenkatalog
Da das Brillensortiment von Brille24 sehr groß ist, und die Eigenmarken alleine schon einen Anteil von 1200 Brillen ausmachen, wurde für die Entwicklung des Brillenberaters nur ein kleiner Katalog ausgewählt. Dieser besteht aus 80 Eigenmarken-Brillen, die im Jahr 2018 zu den Topsellern gehörten. Diese Einschränkung wurde getroffen, da eine Aufbereitung der gesamten Menge an Brillen für den Projektumfang zu zeitintensiv gewesen wäre. Bei der Auswahl wurde darauf geachtet, dass von jeder möglichen Brillenform und Rahmenart mindestens eine Brille vorhanden ist. Für den Zweck der Evaluation und Abschlusspräsentation wurde der Brillenkatalog vorläufig auf 35 Brillen reduziert, und die Auswahl an Brillenmodellen bestellt. Dadurch wurde die Möglichkeit geschaffen, bei der Präsentation des Brillenberaters den Anwendern die Brillenempfehlungen auch real zeigen zu können. Langfristig ist es aber das Ziel, den Brillenkatalog für die Beratung auf das gesamte Sortiment zu erweitern.

Brillenattribute
In der Datenbank von Brille24 waren zu den ausgewählten Eigenmarken-Brillen bereits einige beschreibende Attribute gepflegt. Diese Grundlage wurde genutzt und an die Projektbedürfnisse angepasst und erweitert. Im Zuge der Überarbeitung sind folgende Attribute, durch die eine Brille beschrieben werden kann, festgelegt worden:

- Rahmenart
- Material

- Geschlecht
- Fassungsart
- Glasgröße
- Brillenform
- Tonart
- Farbe

Für die Attribute wurden zusätzlich noch die möglichen Ausprägungen definiert. Die Rahmenart einer Brille wird, wie Abbildung 34 zeigt, in Vollrand, Halbrand und Rahmenlos unterteilt. Beim Material einer Brille wird im Sortiment von Brille24 zwischen Kunst-

Abbildung 34: Ausprägungen der Rahmenart

stoff, Metall, Titan, Edelstahl und Holz unterschieden. Die Eigenmarken-Brillen gibt es allerdings nur aus den ersten drei Materialien, weshalb im Katalog für den Brillenberater nur nach diesen Ausprägungen unterschieden werden musste. Dabei war allerdings auch möglich, dass eine Brille aus einer Kombination der Materialien besteht. Um mit dem Attribut „Geschlecht“ zu beschreiben, für wen eine Brille geeignet ist, gab es nicht nur die Möglichkeiten männlich oder weiblich, sondern auch neutral um Unisex-Brillen kenntlich zu machen. Bei der Fassungsart und der Glasform wurde die Anzahl der Ausprägungen auf drei Möglichkeiten beschränkt. Die Brillen wurden dabei nach dünner, mittlerer und dicker Fassung und kleinen, mittleren und großen Gläsern unterschieden. Bei der Brillenform gibt es die Unterscheidung zwischen den in Abbildung 35 dargestellten Ausprägungen. Für die Beratung und den zusammengestellten Brillenkatalog wurden allerdings Sportbrillen und die Form „Sonstige“, zu denen z.B. mehreckige Brillen gehören, nicht mit einbezogen.

Die Farbe und Tonart können bei der Bestimmung der jeweiligen Attributsausprägung zusammen betrachtet werden. Für die Farben gibt es dabei 12 Möglichkeiten. Es wird zwischen schwarz, blau, rot, braun, meliert, weiß, lila/pink, grün, silber, grau, gold-gelb und transparent unterschieden. Diese Farben können wiederum einer oder mehreren Tonarten zugeordnet werden. Somit kann der Farbton einer Brille hell, dunkel, kräftig, kühl

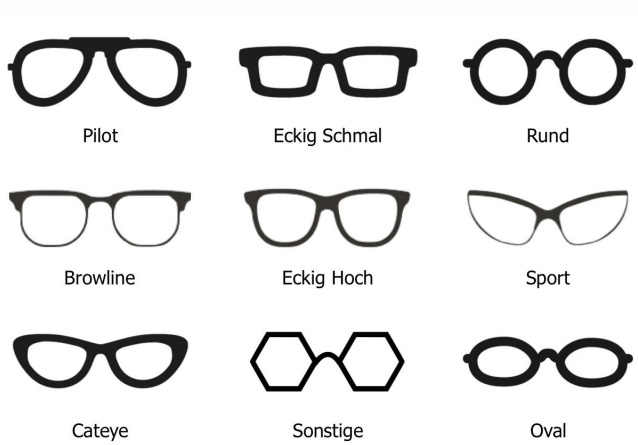


Abbildung 35: Ausprägungen der Brillenform

oder warm sein.

Wie schon bei der Beschreibung des Brillenkatalogs erwähnt, wurde darauf geachtet das von jeder Attributsausprägung im Brillenkatalog etwas vorhanden ist. Die Abbildung 36 zeigt mit welchem Anteil die unterschiedlichen Attributsausprägungen dabei vertreten sind.

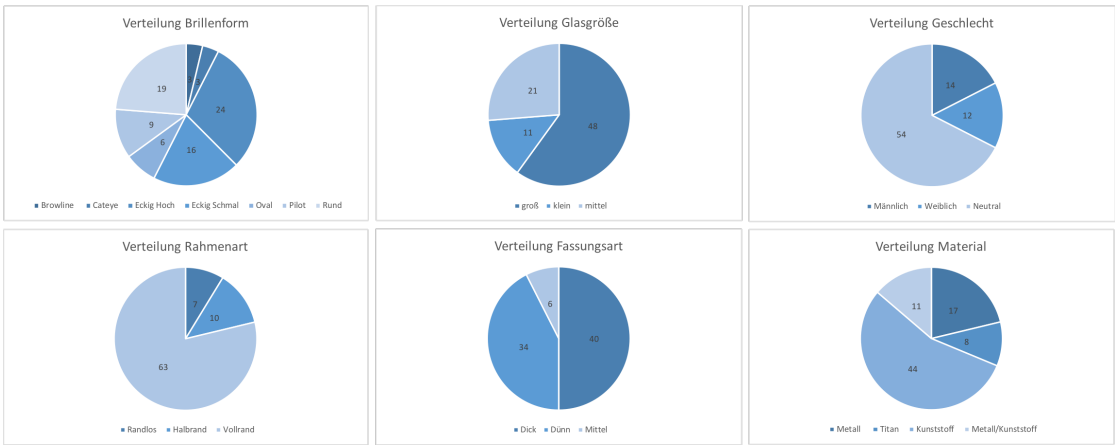


Abbildung 36: Verteilung der Attributsausprägungen

BB.6 Mapping der Merkmale zu den Brillen Nachdem die Brillen des Brillenkatalogs ausgiebig mit beschreibenden Attributen versehen wurden, erfolgte auf dieser Basis die

Erstellung der Regeln. Hierfür wurden die Fashionregeln, die von der Research-Abteilung von Brille24 ausgearbeitet wurden, zur Hilfe genommen. Zunächst fand eine Einarbeitung in diese Regeln statt, um ein tieferes Verständnis für die Aspekte der Brillenberatung zu erlangen. Zusätzlich wurden verschiedene Quellen herangezogen und mit den bisherigen Erkenntnissen von Brille24 verknüpft. Auch hier gab es eine Anpassung auf die projektspezifischen Ziele. So wurden beispielsweise die umfangreichen Fashionregeln auf den Output der künstlichen neuronalen Netze angepasst, indem einige Merkmalsausprägungen weggelassen oder schwerer gewichtet wurden. Gegenübergestellt waren somit die in **??** genannten Attributsausprägungen einer Brille den folgenden Merkmalen mit ihren Ausprägungen:

- Geschlecht: männlich, weiblich
- Gesichtsform: rund, eckig, oval, herzförmig, trapez
- Gesichtsbehaarung: Bart, kein Bart
- Haarfarbe: grau, blond, braun, schwarz, rot, haarlos
- Hautfarbe: sehr hell, hell, oliv, gelbstichig, dunkel

Nachdem das Grundgerüst für die entscheidenden Regeln stand wurden die Brillenattribute den möglichen Ausprägungen der oben gelisteten Merkmale zugeordnet. Bei dieser Zuordnung fand das zuvor angeeignete Wissen Anwendung. Auf diesem Weg ergaben sich die Regeln für das Mapping, um nach der Analyse des Gesichtes durch das Machine Learning passende Brillen empfehlen zu können. Die Regeln sind so aufgebaut, dass zu jeder Gesichtsmerkmalsausprägung passende Brillenattributsausprägungen zugeordnet wurden. Gleichzeitig wurden komplett unpassende Attributsausprägungen ausgeschlossen. Als Beispiel wird die Regel zu einem runden Gesicht im Bezug auf die Brillenform und Fassungsstärke dargestellt:

- passt: eckige Brillenformen (eckig hoch, eckig schmal), Browline, dünne und mittlere Fassungsstärke
- passt nicht: rundliche Brillenformen (rund, oval, Cateye), dicke Fassungsstärke

Daraus ergibt sich somit ein Teil der Regel für runde Gesichter, keine rundlichen Brillen, sondern eckige Modelle zu empfehlen. In den entstandenen Regeln gibt es Kombinationen, die einen stärkeren Einfluss auf die Brillenempfehlung haben, als andere. Dies wurde beim Mapping berücksichtigt, indem ein Ranking der Ausprägungen, die zur Filterung des Brillenkataloges dienen, erstellt wurde.

9.7 Abnahme

Folgend wird die Erfüllung der einzelnen Anforderungen bewertet. Erfüllte Anforderungen werden mit **Abgenommen**, nicht erfüllte Anforderungen mit **nicht Abgenommen** markiert. Anforderungen, die auf andere Art und Weise gelöst wurden erhalten die Markierung **nicht umgesetzt**. Die Übersicht der Abnahme ist in Tabelle 18 dargestellt.

Nr.	Beschreibung	Abgenommen
BB.1	Klassifizierung des Geschlechts	**Abgenommen**
BB.2	Bestimmung des Alters der betrachteten Person	**Abgenommen**
BB.3	Klassifizierung von Bartträgern	**Abgenommen**
BB.4	Klassifizierung von Haar- und Hautfarbe	**Abgenommen**
BB.5	Bestimmung der Gesichtsform	**Abgenommen**
BB.5.1	Bestimmung von Facial Landmarks	**Abgenommen**
BB.5.2	Klassifizierung der Gesichtsform	**Abgenommen**
BB.6	Mapping der Merkmalen zu den Brillen	**Abgenommen**

Tabelle 18: Abnahme der Anforderungen an den Brillenberater

9.8 Fazit

Abschließend kann festgestellt werden, dass alle Anforderungen an den Brillenberater umgesetzt wurden. Mit dem Brillenberater wurde ein Service implementiert, der dem Kunden, auf Basis eines aufgenommenen Fotos, eine Auswahl passender Brille vorschlagen kann. Grundlage für diese Empfehlung bilden eine Menge an Merkmalen, die aus dem Foto extrahiert werden. Diese Merkmale umfassen das Geschlecht, das Alter, die Gesichtsform, Haar- und Hautfarbe sowie die Information, ob es sich um einen Bartträger handelt. Den Abschluss dieses Beratungsprozesses bilden drei Brillen, welche dem Kunden vorgeschlagen werden.

9.9 Ausblick

Die Entwicklung des Brillenberaters im Rahmen dieses Projektes ist abgeschlossen und der Dienst kann bereits eingesetzt werden, um das Einkaufserlebnis der Kunden zu erhöhen. An dieser Stelle sollen noch Empfehlungen gegeben werden, an welchen Stellen der Brillenberater vom Praxispartner weiter optimiert werden kann.

Einen möglichen Ansatzpunkt für Verbesserungen stellt die Klassifizierung der Gesichtsform dar. Der implementierte Ansatz in diesem Projekt liefert die besten Ergebnisse, wenn

ein Nutzer, wie bei einem Passfoto, ohne posieren oder Grimassen, gerade in die Kamera sieht. Damit garantiert wird, dass die nachfolgenden Modelle eine gute Datenbasis erhalten, kann eine sogenannte Head-Pose-Estimation entwickelt werden. Bei diesem Ansatz wird die Pose des Nutzers auf dem Selfie klassifiziert und falls das Bild nicht den Anforderungen entspricht, wird der Kunde aufgefordert, ein neues Bild hochzuladen.

Daneben kann der Ansatz mit den Facial Landmarks weiterverfolgt werden. Die Zielsetzung ist dann allerdings nicht mehr die Klassifizierung der Gesichtsform, sondern das Erfassen der absoluten Maße des Gesichtes. Im aktuellen Ansatz bewertet der Algorithmus bei der Entscheidungsfindung nur, ob das Modell modisch zum Gesicht des Kunden passt. Die Größe der Brille bleibt dabei unberücksichtigt und es erfolgt gelegentlich eine Empfehlung für eine Brille, welche für den Kunden entweder zu groß oder zu klein ist.

Ein dritter Ansatz besteht darin, weitere Merkmale zu klassifizieren. So könnte beispielsweise auch die Farbe der Augen einen Einfluss auf die Empfehlung einer Brille haben.

10 Trendanalyse

Das Teilprojekt Trendanalyse soll zur Unterstützung des Einkaufs bei Brille24 dienen. Dabei liegt der Fokus darauf neue und bestehenden Trends in sozialen Netzwerken zu finden, um gezielt Brillenmodelle bestellen zu können.

Zurzeit müssen die Angestellten manuell die verschiedenen sozialen Medien nach neuen Trends durchsuchen. Dies soll durch die Entwicklung eines Crawlers und die Einführung eines Dashboards, auf dem die verschiedenen Trends übersichtlich angezeigt werden, ersetzt werden. Der Crawler dient dazu, die verschiedenen sozialen Medien auf Basis bestimmter Kriterien zu durchsuchen und die entsprechenden Bilder und die dazugehörigen Metadaten in der Datenbank abzuspeichern. Mittels Maschine Learning sollen dann zusätzliche Informationen zu den verschiedenen Bilddaten erkannt und entsprechend abgespeichert werden. Das Dashboard kann dadurch auf die in der Datenbank hinterlegten Bilder und Informationen zugreifen und diese auswerten und visualisieren.

10.1 Crawler

Eine der Komponenten der Social-Media-Trendanalyse ist der Crawler, der auf Grundlage verschiedener Kriterien soziale Netzwerke nach Beiträgen durchsucht und entsprechende Inhalte, also Bild und Metadaten, herunterlädt und in der Datenbank abspeichert. Dem Crawler muss dabei vorgegeben werden, welche Seiten er durchlaufen und auf welche Elemente er achten soll. Dabei kann sich der Crawler sowohl an dem HTML-Inhalt der Seite als auch an deren XML-File orientieren. Zusätzlich gäbe es auch die Möglichkeit nicht gezielt auf Elemente einer Webseite zuzugreifen sondern den kompletten Inhalt davon zu laden. Hierbei besteht allerdings der Nachteil, das viele unnötige Elemente mit geladen und spätere Analysen benötigt werden würden. Daher wird im Folgenden zunächst das Crawling-Konzept vorgestellt, bei dem Kriterien und das Verhalten des Crawlers festgehalten wurden. Daraufhin werden die Anforderungen geschildert und die konkrete Umsetzung erläutert.

10.1.1 Crawling-Konzept

Das Crawling-Konzept für die Social-Media-Trendanalyse setzt sich aus vier Hauptbestandteile zusammen. Dabei handelt es sich um die allgemeine Umsetzung, Plattform, Suchkriterien und den Output.

Umsetzung

- Zyklus
- Zeitpunkt
- Umsetzung

Bei der Umsetzung geht es darum, in welchem Rahmen der Crawler realisiert und eingesetzt werden soll. Die technische Umsetzung des Crawlers erfolgt dabei mit Selenium und Python. Die genaue Realisierung wird in Abschnitt 10.1.3 beschrieben. Der Zyklus gibt an, in welchem Abstand der Crawler Daten von der Plattform ziehen soll. In diesem Rahmen wurde festgelegt, dass der Crawler kontinuierlich aktiv sein soll um einen möglichst aktuellen Stand der Daten zu bieten. Mit dem Zeitpunkt ist gemeint, im welchem Zeitraum die Beiträge zurückliegen dürfen. Dieser Zeitpunkt wird dem Crawler in Form einer Jahreszahl als Parameter übergeben. Nach dem initialen Crawling wird der letzte Crawlingzeitpunkt abgespeichert und nur aktuellere Beiträge betrachtet.

Plattformen

- Instagram
- Pinterest
- Twitter
- Marken
- Google

Unter diesem Punkt werden die Plattformen aufgeführt, von denen der Crawler die Daten bekommt. Das Hauptaugenmerk liegt dabei aktuell auf Instagram. Interessant wären allerding noch Pinterest und die Google-Bilder-Suche, um die Ergebnisse von Sucheingaben wie „Brillentrends 2018“ zu crawlen. Inwiefern Twitter Anwendung finden wird ist noch ungeklärt.

Suchkriterien

- Marken(Brille24 Marken, auch andere Marken die aufgenommen werden könnten)
- Hashtags

- Influencer/Blogger
  - Nationalität
  - Follower
  - Geschlecht
  - Alter
- Begriffe für Plattformen ohne Hashtags

Die Suchkriterien werden in vier Gruppen zusammengefasst und sind plattformabhängig. Bei dem Kriterium „Marken“ wird sich auf die Marken konzentriert, die Brille24 im Bestand führt. Zusätzlich sind aber auch andere Marken, die vermehrt Onlinepräsenz aufweisen, von Interesse, die eventuell in das Sortiment aufgenommen werden könnten. Hashtags als Suchkriterium sind hauptsächlich für Instagram relevant. Diese beziehen sich größtenteils auf das Thema Brille oder Marken und sollen die aktuellen Brillen der breiten Masse widerspiegeln. Mit den Influencern/Bloggern, sollen dem Crawler, neben der breiten Masse, noch konkrete Personen übermittelt werden, anhand derer Trends festgemacht werden könnten. Einzelne Kriterien, nach denen die Influencer/Blogger hierfür ausgewählt werden sind die Nationalität, die Anzahl der Follower, das Geschlecht und Alter. Besonders von Interesse sind dabei natürlich Brillenträger oder Profile, auf denen viele Brillen von der Person präsentiert werden. Da nicht auf allen Plattformen Hashtags verwendet, erfolgt die Suche über die Eingabe festgelegter Begriffe. In allen Fällen sollen aktuelle Brillentrends ermittelt werden.

Output

- Bild/Bilder(aus Album)
  - Likes
  - Ort
  - Titel
    - Hashtags
  - Upload-Datum
  - Account
  - Markierte Accounts
- Profilinformationen(Influencer)

  - Abonnenten/Follower
  - (Biografie)
  - (richtiger Name)

- Downloaddatum/Zeitstempel

Als Ergebnis des Crawlings soll bestimmter Output erhalten werden. In jedem Fall sollen Bilddaten und das Downloaddatum gespeichert werden. Zu einem gecrawlten Bild bzw. Bilderalbum gehören Upload-Datum, Account und Likes. Optional können auch Titel und Ort vom Nutzer angegeben werden, wobei in dem Titel die Hashtags mit enthalten sind. Diese werden dann ebenfalls abgelegt. Für den Fall, dass nach einem Influencer/Blogger gecrawlt wird, können zusätzlich die Profilinformationen des Accounts herausgelesen werden. Vom hohen Interesse ist dabei nur die Anzahl der Abonnenten.

10.1.2 Anforderungen

Nr.	Beschreibung
CRA 1.1	**Zugriff auf Plattform mit Bildfokus** Der Crawler soll auf eine Plattform, bei der der Fokus auf Bildinhalten liegt, zugreifen können.
CRA 1.2	**Filtern der Beiträge nach bestimmten Suchkriterien** Auf der Plattform soll der Crawler nach bestimmten Accounts oder Hashtags suchen und die dazugehörigen Beiträge finden.
CRA 1.3	**Aufruf der Beiträge auf der Plattform** Der Crawler soll Beiträge der Plattform öffnen und auf die enthaltenden Bild- und Metadaten zugreifen können.
CRA 1.4	**Zeitliche Begrenzung der zu sammelnden Beiträge** Der Crawler soll nur Beiträge berücksichtigen, die in einem zuvor festgelegten Zeitraum veröffentlich wurden.
CRA 1.5	**Eine Dopplung der Beiträge soll weitestgehend ausgeschlossen werden** Der Crawler soll überprüfen, zu welchem Zeitpunkt er zuletzt auf Beiträge der Plattform zugegriffen hat und nur neuere Beiträge beachten.
CRA 1.6	**Speichern der vorher definierten relevanten Daten eines Beitrages in einer JSON-Datei** Der Crawler soll die Bild- und Metadaten für jeden Beitrag in einer JSON-Datei speichern.
CRA 1.7	**Weiterleiten der Daten im JSON-Format an die Datenbank** Der Crawler soll jede entstehende JSON-Datei in die Datenbank schreiben.

10.1.3 Umsetzung

Bei dem Crawler handelt es sich um ein Python-Skript, welches durch die Einbindung von PhantomJS einen headless Browser auf dem Server nutzt. Mit Hilfe des Frameworks Selenium wird virtuell ein Nutzer simuliert und die angegebenen Seiten können durchsucht werden.

Zu Beginn der Umsetzung wurde Firefox als Browser verwendet, um einen Überblick über die benötigten Webelemente und eine visuelle Bestätigung für die richtige Implementierung der entwickelten Funktionen zu bekommen. Dies ermöglichte das grobe Testen des Crawlers. Allerdings kann serverseitig kein Browser verwendet werden, da es keine Benut-

zeroberfläche gibt, auf der der Browser gestartet werden könnte. Demzufolge wurde der Webkit-Browser PhantomJS eingeführt, der es ermöglicht, die Anwendung headless und mittels Script gesteuert auf dem Server zu nutzen.

Um in dem Browser zu navigieren und Funktionen der Elemente auszuführen wird Selenium verwendet. Die geläufigsten Anwendungsfälle von Selenium finden sich in dem Bereich der automatisierten Softwaretests von Webanwendungen. Dies ist der Fall, da Selenium die Möglichkeit bietet, Webelemente auf einer entsprechenden Webseite mittels bspw. des Klassennamens zu finden und mit den gefundenen Elementen zu interagieren. Hierdurch kann dem Browser vorgetäuscht werden, dass er die Eingaben von einem echten Nutzer erhält, der sich durch die Seiten navigiert. Zudem können die Tests beliebig oft und automatisiert ablaufen, weswegen sich das Framework ideal für Softwaretests im Bereich der Webanwendungen anbietet.

Im Anwendungsfall der Trendanalyse wird Selenium für die konkrete Interaktion mit den Funktionselementen der Social-Media-Plattform genutzt. Durch die Nutzung von Selenium werden verschiedene Sicherheitsmechanismen umgangen, die Crawler abwehren würden, da ein realitätsnahes Nutzerverhalten simuliert wird.

Bei der praktischen Umsetzung des Crawlers wurde auf die Erfüllung der zuvor definierten Anforderungen geachtet. Diese werden im Folgenden näher betrachtet.
Die Anforderung *CRA 1.1* wird mittels Selenium erfüllt. Dafür ist die URL der gewünschten Plattform im Code fest implementiert. Diese URL wird dann erweitert durch ein übergebenes Suchkriterium, wie Accountnamen oder Hashtag. Dadurch wird die zugehörige Seite geöffnet und der Crawler erhält Zugriff. Die Suchkriterien werden in einer Liste festgehalten, die in der Datenbank hinterlegt ist. Diese Liste wird der Reihenfolge nach ausgelesen, wodurch Anforderung *CRA 1.2* als erfüllt gilt.
In dieser Liste wird zusätzlich zu jedem Suchkriterium ein Zeitstempel hinterlegt, der den Zeitpunkt des letzten Crawlings darstellt. Dadurch kann sicher gestellt werden, dass keine Beiträge doppelt betrachtet und in der Datenbank abgespeichert werden, welches Anforderung *CRA 1.5* entspricht.
Für das initiale Crawling jedes Suchbegriffs wird außerdem ein Jahr übergeben, welches festlegt, wie weit die zu crawlenden Beiträge in der Vergangenheit liegen dürfen. Posts, die vor dem 1. Januar des übergebenen Jahres veröffentlicht wurden, werden nicht geöffnet und gespeichert, was Anforderung *CRA 1.4* widerspiegelt.
Auf der Übersichtsseite des Suchbegriffs wird der aktuellste Beitrag angeklickt und geöffnet.

Dieser Detailansicht können alle zugehörigen Bilder sowie Metadaten, wie beispielsweise das Uploaddatum und die Likesanzahl, des Beitrags entnommen werden. Durch das Weiterklicken auf den nächsten Beitrag navigiert sich der Crawler durch die Posts und erfüllt somit Anforderung *CRA 1.3*.
Die in dem Crawling-Konzept definierten Bild- und Metadaten können von dieser Detailansicht in einem Python-Objekt abgespeichert werden. Dieses wird anschließend in ein JSON-Objekt überführt, welches dann der Datenbank übermittelt wird. Dadurch sind Anforderung *CRA 1.6* und *CRA 1.7* erfüllt.

10.1.4 Abnahme

Folgend wird die Erfüllung der einzelnen Anforderungen bewertet. Erfüllte Anforderungen werden mit **Abgenommen**, nicht erfüllte Anforderungen mit **nicht Abgenommen** markiert. Anforderungen die auf andere Art und Weise gelöst wurden erhalten die Markierung **nicht umgesetzt**.

Nr.	Beschreibung	Abgenommen
CRA.1.1	Zugriff auf Plattform mit Bildfokus	**Abgenommen**
CRA.1.2	Filtern der Beiträge nach bestimmten Suchkriterien	**Abgenommen**
CRA.1.3	Aufruf der Beiträge auf der Plattform	**Abgenommen**
CRA.1.4	Zeitliche Begrenzung der zu sammelnden Beiträge	**Abgenommen**
CRA.1.5	Eine Dopplung der Beiträge soll weitestgehend ausgeschlossen werden	**Abgenommen**
CRA.1.6	Speichern der vorher definierten relevanten Daten eines Beitrages in einer JSON-Datei	**Abgenommen**
CRA.1.7	Weiterleiten der Daten im JSON-Format an die Datenbank	**Abgenommen**

Tabelle 19: Abnahme der Anforderungen an den Crawler

10.2 Dashboard

Das Dashboard dient der Visualierung und Verknüpfung der Ergebnisse des Crawlers und des Machine Learnings. Diese sollen genutzt werden um die Trends in den sozialen Medien herauszustellen und in aufbereiteter Form darzustellen.

10.2.1 Anforderungen

Nr.	Beschreibung
DAS 1.1	**Ergebnisdarstellung aus Crawler und ML** Das Dashboard soll die Ergebnisse aus Crawler und ML darstellen.
DAS 1.2	**Darstellung durch Grafiken** Das Dashboard soll zur Darstellung unterschiedliche Grafiken nutzen.
DAS 1.3	**Galerie für Bilddaten** Das Dashboard soll Galerien enthalten, in denen die Bildbasis für die Ergebnisse hinterlegt wird und einsehbar ist.
DAS 1.4	**Sinnvolle Ergebnisverknüpfung auf einzelnen Seiten** Das Dashboard soll zur Übersichtlichkeit mehrere Unterseiten enthalten, die Verknüpfungen in einem bestimmten Kontext darstellen.

10.2.2 Konzept

Um eine erste Vorstellung für die Visualisierung und den Umfang des Dashboards zu bekommen, wurden Mockups angefertigt. Diese wurden zu Beginn händisch erstellt, um Änderungen schnell einzeichnen zu können. Hierbei wurde die Überlegung getroffen, das Dashboard mit 4 Unterseiten zu versehen. Es wurde eine Hauptseite erstellt, die eine kurze Übersicht über die Unterseiten enthält und von der aus man zu diesen navigieren kann. Die Navigation kann zudem auch jederzeit von einer Navigationsbar an der linken Seite erfolgen. Im Anschluss wurden diese Mockups mit SAP.BUILD in einem klickbaren Prototypen umgesetzt.

Für die Darstellung der Trends im Dashboard werden sowohl Daten aus dem Crawler als auch des Machine Learnings hinzugezogen. Hierbei handelt es sich um folgende Daten:

- Crawler-Daten
  - Bilder
  - Titel und Hashtags
  - Likes
  - Datum
  - Account
  - Follower, bei Influencern/Markenaccounts
  - optionale Angaben wie Ort und markierte Accounts

- ML-Daten
  - Brillentyp (Sonnenbrille, Korrektionsbrille)
  - Rahmenart
  - Rahmenfarbe
  - Brillenform
  - Geschlecht (und Alter) des Brillenträgers

Für die Darstellung der Trends werden vier Dashboard-Seiten benötigt. Zusätzlich hierzu soll durch eine Konfigurationsseite die Möglichkeit geboten werden, Anpassungen bzgl. der Suchkriterien des Crawlers vorzunehmen. Eine weitere Ergänzung zu den Trendseiten ist eine Einzelseite, die als Galerie dient. Die einzelnen Seiten, deren Bestandteile und mögliche Filterfunktionen werden im Folgenden näher beschrieben und mit Mockups veranschaulicht.

Galerieseite

Die Galerieseite soll als Startseite dienen und für die Darstellung der Bilddaten genutzt werden. Von jeder Seite aus soll dem Nutzer über eine Verlinkung die Möglichkeit geboten werden auf die Galerieseite zu gelangen. Die Bilddaten sollen dabei je nach Ausgangsseite, von der auf die Verlinkung geklickt wird, angepasst werden.

Konfigurationsseite

Die Seite für die Konfiguration soll die in der Datenbank hinterlegten Accounts zu Marken und Influencern und die Hashtags mit Brillen- und Markenkontext abbilden, die für die Filterung der Beiträge in sozialen Medien genutzt werden. Zudem soll hierüber die Möglichkeit geboten werden, Anpassungen vorzunehmen wie z.B. das Löschen und Hinzufügen von Marken und Hashtags. Eine ungefähre Darstellung der Seite ist in Abbildung 37 zu sehen.

Markenseite

Die Seite für die Marken soll die Möglichkeit bieten, Trends mit Bezug zu einer spezifischen Marke darzustellen. Die Auswahl der Marke soll, wie in Abbildung 38 dargestellt, über eine Filterfunktion ermöglicht werden. Nach dem Auswählen wird die Seite mit Inhalt gefüllt. Dabei wird unterschieden zwischen Trends die sich auf Beiträge des Markenaccounts beziehen und Beiträge, die durch ein Hashtag auf die Marke verweisen. Die Ergebnisse zu den Hashtags spiegeln die Trends der breiten

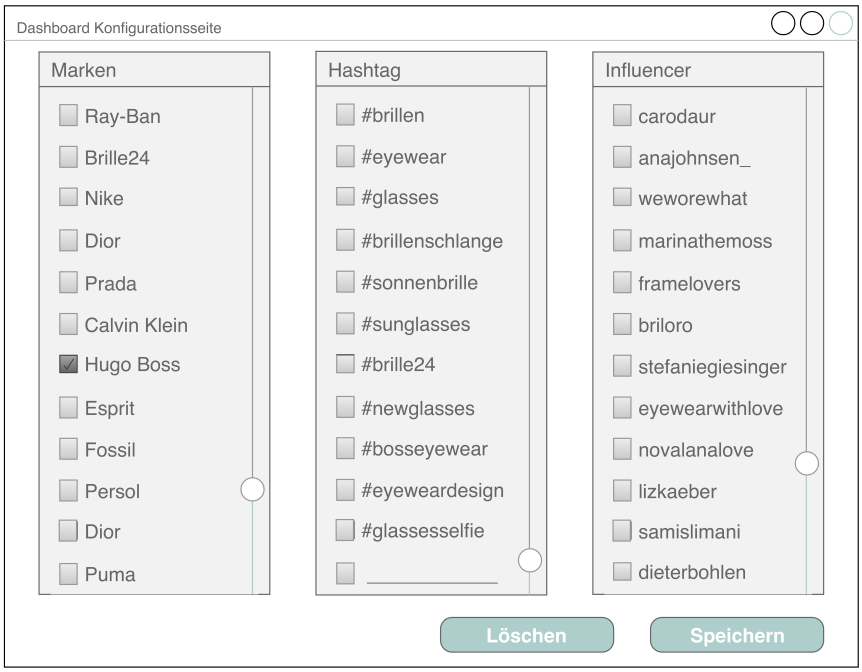


Abbildung 37: Mockup: Konfiguarationsseite

Masse wieder. Sowohl für den Markenaccount als auch für die Hashtag-Beiträge werden die durchschnittlichen Likes pro Beitrag angegeben. Für den Markenaccount wird zusätzlich die Followeranzahl angezeigt und zu den Hashtag-Ergebnissen die Anzahl unterschiedlicher Profile, von denen die Beiträge zu den Hashtags stammen. Für beide Seiten soll jeweils eine Übersicht für die Verteilung der erkannten Brillentypen, das Geschlecht der Brillenträger in den Beiträgen, die Brillenformen, Rahmenarten und -farben gegeben werden. Dabei wird auf die Visualisierung mit Hilfe unterschiedlicher Diagrammtypen und Icons zurückgegriffen.

Seite für die Rahmenfarbe

Auf der Seite für die Rahmenfarbe soll die Entwicklung der Farbtrends in den sozialen Medien wiedergespiegelt werden. Diese sollen mittels Line-Charts sowohl für die gesamten Beiträge als auch einzeln für Influencer/Marken und die breite Masse visualisiert werden. Auf dieser Seite sollen auch die begehrtesten Farben in relativer Häufigkeit dargestellt werden und um die absolute Anzahl ergänzt werden. Ein Mockup für diese Dashboardseite ist in Abbildung 39 zu sehen. Als weitere Möglichkeit wäre es, wenn die notwendigen Ergebnisse von dem Machine Learning geliefert werden können, denkbar, hier die aktuellsten Farbkombinationen darzustel-

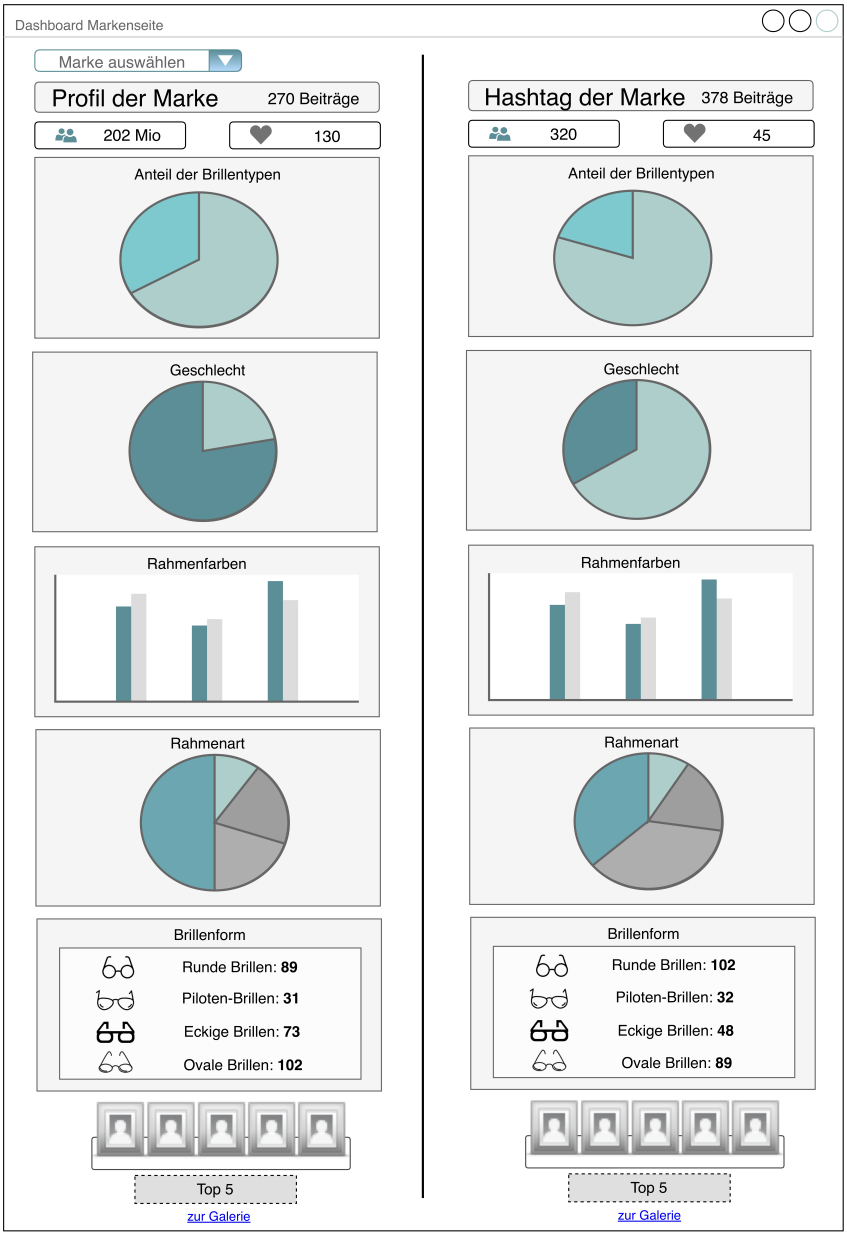


Abbildung 38: Mockup: Markenseite

len.

Seiten für die Rahmenart und Brillenform

Auf der Seite für die Rahmenart sollen aktuelle Trends bzgl. der drei Rahmenarten Vollrand, Halbrand und Randlos abgebildet werden. Die Seite für die Brillenform

Abbildung 39: Mockup: Seite für die Rahmenfarbe

beinhaltet die Trends zu verschiedenen Formen von Brillen. Auf den Seiten soll herausgestellt werden, mit welchem Anteil die einzelnen Rahmenarten und Brillenformen in den sozialen Medien vertreten sind. Der Aufbau dieser Seiten ist identisch, weshalb nur ein Mockup für die Brillenform erstellt wurde, welches in Abbildung 40 gezeigt wird. Die absoluten und relativen Zahlen sollen visualisiert und die zeitliche Entwicklung veranschaulicht werden. Aufgeteilt werden die Trends, wie bei der Seite für die Rahmenfarbe, nach der Gesamtheit der Beiträge, Influencern/Marken und der breiten Masse.

Filterfunktion auf Seiten für Rahmenfarbe/-art und Brillenform

Abbildung 40: Mockup: Seite für die Brillenform

Auf jeder dieser Seiten soll die Möglichkeit gegeben sein, die Ergebnisse nach Geschlecht, männlich, weiblich und neutral, sowie Brillentyp, Sonnen- oder Korrektionsbrille, zu filtern. Zusätzlich kann auf jeder Seite nach den Attributen der zwei anderen Seite gefiltert werden, z. B. auf der Seite für die Brillenform nach Rahmenfarbe und -art.

Allgemein

Auf allen Seiten wird der Betrachtungszeitraum der Beiträge und die absolute Anzahl an Beiträgen mit angegeben. Zudem sollen Teile der Datengrundlage in Form einer Galerie sichtbar gemacht werden, auf durch eine Verlinkung am Ende jeder Seite zugegriffen werden kann. Oberhalb der Verlinkung werden die TOP 5, das heißt die fünf Bilder mit den meisten Likes, für die Influencer/Marken und die breite Masse

durch Vorschaubilder abgebildet. Zu jedem Bild werden zusätzlich die relevanten Metadaten angezeigt, um so z. B. neue Persönlichkeiten für Kooperationen zu finden.

10.2.3 Umsetzung

Um einen Grundbaustein für das Dashboard zu legen, haben wir uns für das FUSE Template entschieden. Dabei handelt es sich nicht um ein klassisches Template, sondern um eine Applikation die auf Angular basiert und in Typescript geschrieben wurde. Zudem verwendet es Googles Material Design. Zusätzlich mussten verschiedene Bibliotheken eingebunden werden. Chart.js wurde verwendet um die unterschiedlichen Diagramme darzustellen. Zur Darstellung der Bildergallerien wurde die Ngx-Gallery eingebunden. Durch die zusätzliche Verwendung von font-awesome können benötigte Icons dargestellt werden.

Zum Befüllen der einzelnen Seiten und dazugehörigen Diagramme wurde eine Schnittstelle zum Backend auf die Datenbank benötigt, um auf die vom Crawler gesammelten Bilder und dazugehörigen Daten sowie die zusätzliches Informationen, die durch das Machine Learning ausgewertet wurden, zuzugreifen und somit die Anforderung DAS 1.1 zu erfüllen. Dabei wurde eine View in CouchDB erstellt, die alle Bilder enthält, die als vom ML analysiert markiert wurden. Bei der Abfrage auf die Datenbank wird dabei der gewünschte Betrachtungszeitraum eingegrenzt, damit keine älteren und somit für den Zeitraum irrelevanten Daten mitgeschickt werden müssen. Die Daten sind dabei ebenfalls nach Aktualität sortiert, damit eine chronologische Auswertung für die Diagramme im Frontend vorgenommen werden kann und die aktuellsten Bilder in der Galerie an erster Stelle zu sehen sind, wie in DAS 1.2 und DAS 1.3 beschrieben. Die Anfrage an das Backend wird zudem nur beim aktualisieren des Dashboards gestartet. Die Bilder und Informationen bleiben beim Navigieren durch die einzelnen Komponenten gespeichert, damit keine unnötige Performance beim wechseln der Seite verschwendet wird, wodurch auch DAS 1.4 erfüllt werden kann. Lediglich die Selektierung auf Grundlage der gewählten Filter und Kriterien der gewünschten Seite führen zu kleineren Ladezeiten.

10.2.4 Abnahme

Folgend wird die Erfüllung der einzelnen Anforderungen bewertet. Erfüllte Anforderungen werden mit **Abgenommen**, nicht erfüllte Anforderungen mit **nicht Abgenommen** markiert. Anforderungen die auf andere Art und Weise gelöst wurden erhalten die Markierung **nicht umgesetzt**.

Nr.	Beschreibung	Abgenommen
DAS.1.1	Ergebnisdarstellung aus Crawler und ML	**Abgenommen**
DAS.1.2	Darstellung durch Grafiken	**Abgenommen**
DAS.1.3	Galerie für Bilddaten	**Abgenommen**
DAS.1.4	Sinnvolle Ergebnisverknüpfung auf einzelnen Seiten	**Abgenommen**

Tabelle 20: Abnahme der Anforderungen an das Dashboard

10.3 Datenstruktur

Dieser Abschnitt enthält die Definition der Datenstruktur für die Trendanalyse zur Erfüllung aller erarbeiteten Anforderungen für dieses Teilprojekt. Dafür wird zunächst die Problematik erläutert, die Anforderungen an die Datenstruktur ausformuliert und Umsetzung beschrieben.

10.3.1 Beschreibung

Bei der Trendanalyse werden Bilder und dazugehörige Daten von einem Crawler gesammelt. Die gespeicherten Bilder werden daraufhin mittels Machine Learning analysiert. Dabei wird geprüft, ob generell eine Brille auf dem Bild vorhanden ist, sowie Eigenschaften wie bspw. Farbe, Rahmenart etc. der Brille ermittelt.
Damit die gewünschten Daten gespeichert werden können, wird eine Datenstruktur benötigt, die angibt, in welcher Form die Daten in der Datenbank abgelegt werden sollen. Hierbei wird JSON-Schema genutzt, welches von dem Crawler erstellt und mit Daten sowie base64-codierten Bild gefüllt und später vom ML mit weiteren Informationen ergänzt wird.

10.3.2 Anforderungen

Die Anforderungen für die Datenstruktur wurden durch ein Interview mit Brille24 sowie durch bereits bestehende Anforderungen des Crawlers ermittelt. Interviewt wurde dabei eine Mitarbeiterin, die für den Einkauf der Brillen und die Erkundung der sozialen Netzwerke nach neuen Trends verantwortlich ist. Dabei wurde sie gefragt, welche Informationen im Dashboard angezeigt werden sollen, um ihre Recherche nach neuen Trends zu vereinfachen. Dies schließt ebenfalls die Informationen mit ein, die vom ML auf den Bildern erörtert werden. Hieraus ergaben sich folgende Anforderungen für die Datenstruktur, welche in Tabelle 21 zu sehen sind.

Nr.	Beschreibung
DAT.1.1	**Unterscheidung von Hashtag und Influencer** Im Dashboard sollen die gesammelten Information nach Hashtag und Influencer unterteilt werden, um zu erkennen, welche Trends sich bei der breiten Masse abzeichnen und welche von den Influencern vorgelebt werden.
DAT.1.2	**Angabe über den Namen des Accounts** Um zu erkennen, von welchem Account die geladene Brille stammt, soll der Name des Accounts gespeichert werden.
DAT.1.3	**Angaben über die Anzahl der Follower und Likes des Bildes** Damit erkennbar ist, wie populär die Accounts sind und um dementsprechend Rückschlüsse ziehen zu können, wie viele Personen die Posts erreichen, sollen die Anzahl der Follower und die Likes der geladenen Bilder gespeichert werden.
DAT.1.4	**Angaben über den Titel und Ort des Bildes** Um die Möglichkeit zu bieten Hashtags und auch anderes Kontextwissen eines Bildes zu ermitteln, sollen die Titel und Standort der Posts gespeichert werden.
DAT.1.5	**Angabe über das Uploaddatum des Bildes** Damit nachvollziehbar ist, wie alt bzw. aktuell ein Post ist, soll das Uploaddatum gespeichert werden. Dieses kann zudem dabei helfen, wiederkehrende Trends aufzuzeigen.
DAT.1.6	**Markierte Personen des Posts speichern** Da sich häufig unter den markierten Personen eines Posts auch die Marken des Produktes befinden, sollen diese mit gespeichert werden.
DAT.1.7	**Ergebnisse des ML abspeichern** Die Ergebnisse des Machine Learnings sollen erfasst werden können. Diese betreffen Informationen bezüglich Nutzer (Geschlecht) und Brille (Brille oder Sonnenbrille, Brillenform Rahmenfarbe, Brillenart).

Tabelle 21: Anforderungen an die Datenstruktur

10.3.3 Umsetzung

In Abbildung 41 ist ein gecrawltes JSON-Objekt zu sehen, das auch bereits durch das ML analysiert wurde. Zu sehen ist, wie die verschiedenen Anforderungen umgesetzt wurden. Durch das Datenobjekt kann nun ersichtlich werden, ob das Bild von einem Influencer oder über ein Hashtag gepostet wurde, sprich ob es bei einer einzelnen Person oder der breiten Masse anzusiedeln ist, wie in DAT.1.1 beschrieben. Außerdem ist ersichtlich, von wem das Bild hochgeladen wurde und wie viele Follower die Person hat, wie in DAT.1.2 und DAT.1.3 gefordert. Zudem werden Titel, Likes, Uploaddatum und markierte Personen gespeichert, um weitere Informationen über das geladene Bild zu erhalten, wie ebenfalls in DAT.1.3 und zusätzlich in DAT.1.4, DAT.1.5 und DAT.1.6 beschrieben. Zur Vollständigkeit wurde auch das Downloaddatum geladen. Des Weiteren wurde gespeichert, ob sich auf dem Bild mindestens eine Brille befindet, um zu erkennen, ob das Bild für weitere Auswertung durch das ML und für die generelle Darstellung im Dashboard relevant ist. Sollte eine Brille gefunden worden sein, können zudem weitere Informationen über die Brille mittels ML herausgefunden und abgespeichert werden. Dies betrifft die Informationen, die in den Anforderungen erläutert wurden, wie bspw. Farbe, Form und Art bzw. Typ der Brille (DAT.1.9).

10.3.4 Abnahme

Folgend wird die Erfüllung der einzelnen Anforderungen bewertet. Erfüllte Anforderungen werden mit **Abgenommen**, nicht erfüllte Anforderungen mit **nicht Abgenommen** markiert. Anforderungen die auf andere Art und Weise gelöst wurden erhalten die Markierung **nicht umgesetzt**.

Nr.	Beschreibung	Abgenommen
DAT.1.1	Unterscheidung von Hashtag und Influencer	**Abgenommen**
DAT.1.2	Angabe über den Namen des Accounts	**Abgenommen**
DAT.1.3	Angaben über die Anzahl der Follower und Likes des Bildes	**Abgenommen**
DAT.1.4	Angaben über den Titel und Ort des Bildes	**Abgenommen**
DAT.1.5	Angabe über das Uploaddatum des Bildes	**Abgenommen**
DAT.1.6	Markierte Personen des Posts speichern	**Abgenommen**
DAT.1.7	Ergebnisse des ML abspeichern	**Abgenommen**

Tabelle 22: Abnahme der Anforderungen an den Datenstruktur der SMT

```
{
  "_id": "smt-picture-002eb54babc17c9a6da19ee8419e712d",
  "_rev": "1-34e731eceac96ca383929ff438d1cb0c",
  "type": "influencer",
  "account": "dieterbohlen",
  "follower": 1182113,
  "text": "Hallo bei #dieterstagesschau ! Fahrradtour mit Carina... #getoutside #frischerwind #sporteln #anzeige",
  "likes": 0,
  "upload_date": "2018-09-13 14:37:55",
  "download_date": "2019-03-28 17:37:14",
  "location": null,
  "marked": [],
  "sex": "male",
  "sunglass": "sunglass",
  "framecolor": "brown",
  "frameform": "eckig_hoch",
  "frametype": "fullrim",
  "found_glasses": true,
  "analysed": true,
  "_attachments": {
    "picture0.b64": {
      "content_type": "application/octet-stream",
      "revpos": 1,
      "digest": "md5-ULjTPEHHHYW5JpSeCwDkkg==",
      "length": 150968,
      "stub": true
    }
  }
}
```


Abbildung 41: Mockup: Konfiguarationsseite

10.4 Domänenwissen

Für die Erkennung von aktuellen Brillentrends und um zu wissen worauf es dabei ankommt, ist ein spezifisches Branchenwissen notwendig. Hierfür wurde, neben eigenen Recherchen, nah im Austausch mit Brille24 zusammengearbeitet. Mit einer ausgebildeten Augenoptikerin, die als Marketingmanagerin und im Einkauf bei Brille24 tätig ist, wurde das benötigte Wissen in mehreren Meetings erarbeitet. Dabei ging es im Allgemeinen darum, Kenntnisse zu erlangen, die als Grundstein für die Datenset-Erstellung und für das Machine Learning der Trendanalyse dienten. Die Basis wurde dahingehend geschaffen, indem zunächst festgehalten wurde, welche Rahmenarten, Brillenformen und mögliche Farben es überhaupt gibt. Dies war wichtig, um die Brillenbilder der Datengrundlage mit entsprechenden Labeln versehen zu können.

Bei den drei möglichen Rahmenarten (Randlos, Halbrand und Vollrand) fiel eine Zuordnung relativ leicht. Unstimmigkeiten bei der Unterscheidung gab es nur bei Browline-Brillen, bei denen es aufgrund von unscharften Bildern oder ungünstigen Lichtsverhätnisses,

zum Teil schwer zu erkennen war, ob ein dünner Unterrand oder keiner vorhanden. Anders war dies bei den Brillenformen. In diesem Fall gibt es nicht nur drei unterschiedliche Möglichkeiten, sondern die Einteilung sollte in neun verschiedene, mögliche Formen erfolgen. Hierbei wurden folgende Kategorien zusammen mit Brille24 festgelegt:

- Browline-Brillen,
- Pilot-Brillen,
- Cateye-Brillen,
- runde Brillen,
- ovale Brillen,
- eckig, schmale Brillen,
- eckig, hohe Brillen und
- sonstige Brillen.

In der Kategorie „sonstige Brillen“ sind Sportbrillen und ausgefallene Formen, wie herzförmige oder mehreckige Brillen, zusammengefasst. Um qualitativ hochwertige Ergebnisse mit der Datengrundlage erzielen zu können, war eine genaue Absprache für die Zuteilung zu diesen Kategorien erforderlich. Es wurde geklärt, welche Merkmale einer Brille für welche Form entscheidend sind. So war es z.B. wichtig zu wissen, wie der Oberrand einer Brille sein muss, damit sie als Cateye angesehen werden kann oder ab wann eine Brille als eckig hoch und nicht mehr als eckig schmal gilt. Für jede Kategorie wurden also Entscheidungsregeln festgelegt und abgestimmt, die eine Zuordnung erleichtert und vereinheitlicht haben.

Ein ähnliches Vorgehen wurde auch für die möglichen Brillenfarben angewandt. Hier wurde sich an den Farben aus dem Brille24-Onlineshop orientiert. Dieser filtert die Brillen nach den folgenden 12 Farbkategorien:

- Schwarz,
- Blau,
- Rot,
- Braun-Kupfer-Bronze,
- Meliert,

- Weiß,
- Lila-Pink,
- Grün,
- Silber,
- Grau,
- Gold-Gelb-Orange und
- Transparent.

Wie oben zu sehen, werden teilweise mehrere Farben zusammengefasst. So entsprechen z.B. Gelb, Gold, und Orange einer Kategorie. Dies musste für die Aufbereitung der Datengrundlage berücksichtigt werden. Zudem wurde der Umgang mit weiteren Farbtönen, die nicht explizit bei einer Kategorie aufgeführt sind, geklärt, damit ein übereinstimmendes Verständnis bei der Farbzuordnung existieren konnte.

10.5 Machine Learning

Im folgenden Abschnitt werden die Komponenten des Machine Learnings für die Trendanalyse vorgestellt. Das Vorgehen folgt dem bekannten Ablauf, nach dem zunächst die Anforderungen charakterisiert werden und diese anschließend chronologisch bearbeitet werden.

10.5.1 Beschreibung

Die obergeordnete Zielsetzung der Trendanalyse besteht darin, Bilder aus sozialen Netzwerken auszuwerten. Bisher wurden ein Crawler für das Beschaffen der Bilder, eine Datenstruktur zum Ablegen der Daten und ein Dashboard zur Visualisierung der Trends entworfen. Das Machine Learning ist der letzte fehlende Baustein, damit die Trendanalyse eingesetzt werden kann. Es soll die gespeicherten Bilder aufrufen und auswerten. Für diesen Zweck wird eine Pipeline entworfen, welche im folgenden Abschnitt genauer vorgestellt wird. Das Konzept sieht vor, dass die Bilder aus der Datenbank in regelmäßigen Abständen geladen werden und die Machine Learning Komponenten diese Bilder automatisiert auswerten.

10.5.2 Anforderungen

Folgend werden die Anforderungen definiert. Die Anforderungen wurden teilweise durch den Projektpartner Brille24 vorgegeben. Andere Anforderungen wurden in einem gemeinsamen Workshop erarbeitet oder entstanden während der Entwicklung.

Nr.	Beschreibung
SMT-ML.1	**Erkennung einer Brille auf Bild** Auf den Social-Media-Beiträgen, die der Crawler abspeichert, sollen möglichst viele Brillen gefunden und ausgeschnitten werden. Dabei ist zu berücksichtigen, dass False Positive Ergebnisse auch an die folgenden Netze weitergegeben werden und somit die Qualität der Ergebnisse beeinträchtigen.
SMT-ML.2	**Klassifizierung des Geschlechts einer zu erkennenden Person** Das Geschlecht des Brillenträgers soll ermittelt werden.
SMT-ML.3	**Klassifizierung Brille oder Sonnenbrille** Die gefundene Brille soll in die Klassen „Brille“ und „Sonnenbrille“ eingeteilt werden.
SMT-ML.4	**Klassifizierung der Brillenform** Die selektierte Brille soll hinsichtlich ihrer Form analysiert werden. Die Einteilung erfolgt in 8 Kategorien (siehe Abschnitt 10.4).
SMT-ML.5	**Klassifizierung der Rahmenfarbe** Die Rahmenfarbe der Brille wird klassifiziert und soll den internen Farbkategorien von Brille24 zugeordnet werden (siehe Abschnitt 10.4).
SMT-ML.6	**Klassifizierung der Brillenart** Die letzte Einteilung erfolgt hinsichtlich der Brillenart. Dabei wird die Brille den Klassen „randlos“, „halbrand“ und „vollrand“ zugeordnet.

Tabelle 23: Anforderungen an das Machine Learning

Für die Auswertung der Bilder aus den sozialen Netzwerken soll eine Pipeline entworfen werden. Diese Pipeline enthält die Fotos aus der Datenbank als Input und gibt anschließend ein JSON-Objekt mit den Ergebnissen der Analysen zurück. Der Grundgedanke dieser Analysen ist, dass sich Trends im Brillenmarkt durch bestimmte Eigenschaften der Brillen abbilden lassen (siehe Abschnitt 10.4). Diese Eigenschaften umfassen beispielsweise die Rahmenfarbe, die Brillenform und die Brillenart. Durch den Einsatz der Pipeline sollen demnach die gesammelten Bilder im Hinblick auf diese Merkmale untersucht werden. Im

Folgenden wird der Aufbau der Pipeline erläutert.

Wenn ein neues Bild für die Analyse geladen wird, muss zunächst sichergestellt werden, ob der Beitrag überhaupt eine Brille trägt. Zwar wurden in der Vorauswahl der zu durchsuchenden Seiten, Personen und Seiten favorisiert, von denen häufige Beiträge mit Brillenbezug zu erwarten sind, allerdings kann dadurch natürlich nicht sichergestellt werden, dass immer mindestens eine Brille auf dem Bild zu sehen ist. Da außerdem auch mehrere Brillen pro Bild ausgewertet werden sollen, wird zunächst eine Object Detection implementiert, die Brillen auf Bildern ausfindig machen kann. Wenn jetzt auf einem Beitrag eine oder mehrere Brillen vorhanden sind, sollen diese gefunden und ausgeschnitten werden, um dann im nächsten Schritt für die weitere Auswertung bereit zu stehen. Können keine Brillen auf dem Bild erkannt werden, dann soll der Beitrag nicht weiter berücksichtigt werden.

Wenn eine Menge an ausgeschnittenen Brillen erstellt werden konnte, soll im nächsten Schritt die weitere Analyse erfolgen. Dafür müssen Modelle erstellt werden, welche die selektierten Brillen in verschiedene Klassen einteilt. Die Klassifizierungen betreffen die Brillenform, die Rahmenfarbe, die Brillenart und es soll darüber hinaus geprüft werden, ob es sich um eine Sonnenbrille handelt.

Die Anforderungen des Dashboards sehen vor, dass nicht nur die Eigenschaften der Brille erfasst werden. Dem Anwender soll ermöglicht werden, die aktuellen Trends auch nach dem Geschlecht, der abgebildeten Personen filtern zu können. Dementsprechend müssen sowohl eine Gesichtserkennung als auch eine Klassifizierung hinsichtlich des Geschlechts implementiert werden. Wird keine Person auf dem Bild erkennt, sollen auch keine Informationen zu dem Bild abgespeichert werden.

Die gesammelten Anforderungen des Machine Learnings in der Trendanalyse sind in Tabelle 23 zusammenfassend dargestellt.

10.5.3 Labelstrategien und -tools

Im folgenden Abschnitt werden die Strategien für das Labeln der Daten beschrieben, die für die Umsetzung der Social Media Trendanalyse notwendig waren. In diesem Teilprojekt galt es hierfür zum einen eine Object Detection und zum anderen die Erkennung von Trends zu realisieren. Für diese beiden Anwendungsfälle wurden unterschiedliche Vorgehensweisen beim Datenset-Aufbau eingesetzt, die im Weiteren vorgestellt werden.

Object Detection Ziel der Labelstrategie für die Object Detection war es, ein Machine Learning Modell so trainieren zu können, dass es erkennt, ob in einem vorliegenden Bild eine Brille zu sehen ist oder nicht. Ist dies der Fall, soll um das erkannte Objekt eine Bounding Box gelegt werden, damit im späteren Verlauf des Prozesses mithilfe dieser Box, das Bild zurechtgeschnitten und für die weitere Verarbeitung vorbereitet werden kann.

Aus diesem Ziel ergab sich die Voraussetzung, dass ein Datenset aufgebaut werden musste, auf dessen Basis ein Netzwerk trainiert werden kann, welches eine oder mehrere Bounding Boxes um jeweilige Brillenobjekte legt. Um solch ein Datenset zu erstellen, wurde eine Vielzahl an Bildern mit Brillen aus den unterschiedlichsten Perspektiven benötigt. Bei diesen handelte es sich bereits um Bilder aus den sozialen Medien, damit eine möglichst realitätsnahe Datengrundlage geschaffen werden konnte. Die Brillenbilder sollten dann händisch mit Bounding Boxes versehen werden, welche die jeweilige Brille bzw. Brillen umschließen. Als Hilfsmittel wurde hierfür der VGG Image Annotator [8] eingesetzt.

Trenderkennung Für die Erkennung von Trends ist nicht nur ein Machine Learning Modell ausreichend, sondern es sind mehrere notwendig. Dies musste auch bei der Labelstrategie für die Datengrundlage beachtet werden. Ziel war es, eine Grundlage zu schaffen, mit der Modelle für unterschiedliche Klassifizierungsaufgaben trainiert werden können. Konkret wurden Daten benötigt, die eine Klassifikation nach Sonnenbrille/Brille, Rahmentyp und Form einer erkannten Brille ermöglichen und zudem sollte das Erkennen der Rahmenfarbe möglich gemacht werden. Rohdaten, die sich als Basis für das Training der Modelle eigenen, konnten mithilfe des Social-Media Crawlers akquiriert werden. Relevante Ausschnitte der Bilder, also die wo eine Brille zu erkennen ist, konnten mit der Object Detection gewonnen werden und als Grundlage für den Aufbau des Datenpools dienen. Dieser Datenpool wurde entsprechend der betrachteten Anwendungsfälle annotiert und einzelne Datensets aufgebaut.

Brille/Sonnenbrille: Ob es sich bei einer erkannten Brille um eine Sonnenbrille oder einfache Brille handelt kann durch eine einfache binäre Klassifizierung herausgefunden werden. Hierfür wurde ein eigenes Labeltool bzw. ein Script erstellt, mit dem die Brillenbilder aus dem Datenpool mit Annotationen für Brille bzw. Sonnenbrille versehen wurden. Die Vermerke, 0 für Sonnenbrille und 1 für eine normale Brille, wurden für die weitere Verwendung in eine seperate CSV-Datei geschrieben.

Rahmentyp: Für die Unterscheidung des Rahmentyps kamen die erlangten Kenntnisse

[8] http://www.robots.ox.ac.uk/ vgg/software/via/

aus der in Abschnitt 10.4 beschriebenen Absprache und Wissensaneignung mit Brille24 zur Anwendung. Auch hierfür wurde das erstellte Labeltool genutzt. Da es drei mögliche Rahmentypen gibt, wurde es für diesen Anwendungsfall leicht adaptiert. Anstelle von zwei Annotationen konnte anschließend zwischen drei Annotationen gewählt werden, die jeweils für einen Rahmentyp standen, gewählt werden. Die Vermerke, repräsentiert durch 0, 1 und 2, wurden wiederum in einer CSV-Datei abgespeichert.

Brillenform: Auch für die Klassifizierung der Brillenform war das angeeignete Domänenwissen aus 10.4 notwendig. Bei diesem Anwendungsfall musste noch genauer darauf geachtet werden, worauf es bei der Unterscheidung von den verschiedenen Formen ankommt. Zur Vereinfachung des Labelns wurde auch hier auf das Labeltool zurückgegriffen, nachdem eine Anpassung stattfand. Neben den erweiterten Annotationen wurden zusätzlich Icons für jede mögliche Brillenform hinterlegt, um für die Zuordnung der Brillenbilder ein „Muster“ zu haben.

Rahmenfarbe: Nachdem in Absprache mit Brille24 die zwölf möglichen Farben und der Umgang mit anderen Farbtönen geklärt war, sollte mit dieser Basis ein Datenset für die Farbklassifizierung aufgebaut werden. Hierfür wurde im Gegensatz zu den anderen Anwendungsfällen nicht auf das eigene Tool sondern auf den VGG Image Annotator zurückgegriffen. Mit der Möglichkeit des Anlegens von Checkboxen für jede der zwölf Farben, wurde eine Auswahl von Bildern aus dem Datenpool mit Annotationen zur Farbe der abgebildeten Brille versehen.

10.5.4 Umsetzung

Im Folgenden wird beschrieben, wie die definierten Anforderungen umgesetzt wurden.

SMT-ML.1 Erkennung einer Brille auf Bild Die eingehenden Bilder, die aus den sozialen Netzwerken gewonnen werden, können in der Rohfassung noch nicht für die Klassifizierung der Merkmale verwendet werden. Zunächst müssen alle Brillen auf den Fotos ausfindig gemacht und anschließend ausgeschnitten werden. Zur Lösung dieser Anforderung wird eine Object Detection implementiert, welche ein Bild in der Rohfassung als in Input erhält und dann die entsprechenden Bounding Boxes zurückgibt. Der Aufbau solcher Architektur unterscheidet sich deutlich von den bisher bearbeiteten Klassifizierungsaufgaben. Damit diese Anforderung schnell gelöst werden kann, wird auf eine bestehende Implementierung zurückgegriffen, welche mit eigenen Daten neu trainiert werden kann.

Bei der gewählten Architektur handelt es sich um ein RetinaNet, welches mit einem vortrainierten ResNet50-Backbone geladen wird [Lin+17]. Der Code der Implementierung ist bei GitHub frei verfügbar[9].

```
from keras_retinanet import models

model = models.backbone('resnet50').retinanet(num_classes=1)

model.compile(
  loss={
    'regression'    : keras_retinanet.losses.smooth_l1(),
    'classification': keras_retinanet.losses.focal()
  },
  optimizer=keras.optimizers.adam(lr=1e-5, clipnorm=0.001)
)

```

Listing 12: Laden des RetinaNets

In Listing 12 ist dargestellt, wie das Modell geladen wird. Dabei ist zu beachten, dass sich die Loss-Funktion aus mehreren Teilen zusammensetzt und das kompilieren des Modells anders gelöst werden muss. Bei der verwendeten Bibliothek müssen die Hyperparameter nicht angepasst werden, da für die Trainingsroutine ein Skript bereits steht, welches diese Einstellungen übernimmt und das Training beendet, sobald der Loss stagniert. Für das Training des Modells können entweder öffentliche (Pascal VOC, MS COCO) oder selbst erstellte Datasets verwendet werden. In unserem Anwendungsfall werden die selbst annotierten Daten verwendet und es stehen insgesamt 25000 Bilder mit den entsprechenden Bounding Boxes zur Verfügung (siehe Abschnitt 10.5.3). Im Rahmen des Preprocessings wird eine CSV Datei erstellt, welche alle Bildpfade mit den entsprechenden Koordinaten der Bounding Boxes enthält. Zusätzlich verlangt das Skript, dass die Objekte klassifiziert werden. In diesem Anwendungsfall erhält jedes Objekt das Attribut „glasses“. Anschließend wird das Training durchgeführt und ein Beispiel für die finale Anwendung der Object Detection ist in Abbildung 42 dargestellt. Auf den Nutzerbildern aus den sozialen Netzwerken können jetzt Brillen erkannt und ausgeschnitten werden.

[9] https://github.com/fizyr/keras-retinanet

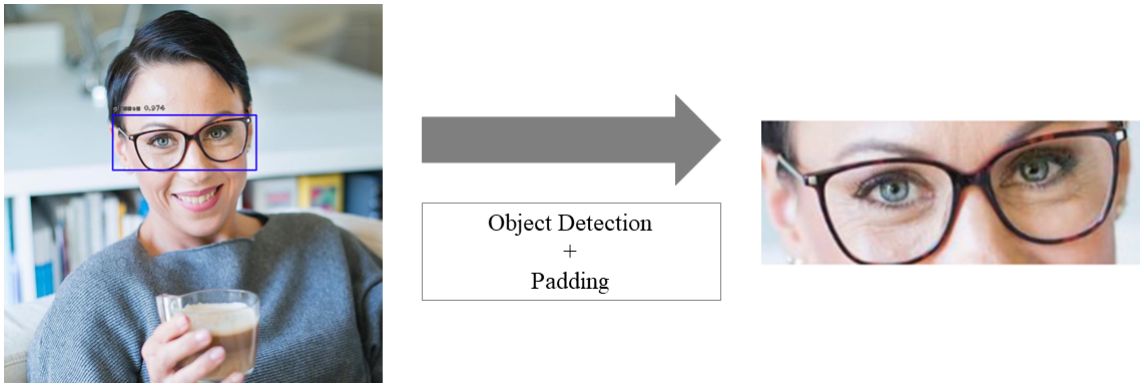


Abbildung 42: Anwendung der Object Detetction

SMT-ML.2 Klassifizierung des Geschlechts einer zu erkennenden Person Um die Trends des Brillenmarktes auch hinsichtlich des Geschlechts der Brillenträger filtern zu können, muss das Geschlecht der Person klassifiziert werden. Dafür wird zunächst eine Face Detection implementiert, die Gesichter erkennt und ausschneidet. Anschließend wird das Geschlecht der gecroppten Gesichter ausgewertet. Für die Face Detection wird dlib verwendet und die Klassifizierung erfolgt durch ein speziell trainiertes MobileNetV2. Beide Anwendungen sind bereits für den Brillenberater umgesetzt worden und können einfach übernommen werden. Damit kann das Geschlecht bei der Analyse berücksichtigt werden und diese Anforderung ist erfüllt.

SMT-ML.3 Klassifizierung Brille oder Sonnenbrille Bei der Unterscheidung zwischen (Korrektions-) Brille und Sonnenbrille handelt es sich um eine binäre Klassifizierung. Wie zuvor beschrieben, wurde hierfür ein eigenes Datenset aufgebaut. Ein großer Vorteil, die Daten selbst zu labeln, besteht darin, dass selbstständig darüber entschieden werden kann, wie die Labelinformationen abgespeichert werden. Dies erleichtert das Preprocessing der Daten erheblich. In diesem Fall wurden die Label mitsamt der zugehörigen Bildpfade in eine CSV-Datei geschrieben. Mithilfe der pandas-Library konnte die CSV-Datei leicht eingelesen und für das Training der Modelle weiterverarbeitet werden. Die Bilder selbst wurden innerhalb der „load batch“-Methode anhand der abgespeicherten Pfade eingelesen. Da alle Bilder mithilfe der Object Detection anhand ihrer Bounding Box zugeschnitten wurden, mussten sie nach dem Laden noch in das richtige Format von 224x224 Pixeln gebracht werden. Da die Bounding Boxes in aller Regel rechteckig waren, wurden die

Bilder mit weißen Pixeln gepaddet, um das benötigte quadratische Format zu erhalten, ohne die dargestellten Brillen zu verzerren (siehe Abbildung 43).

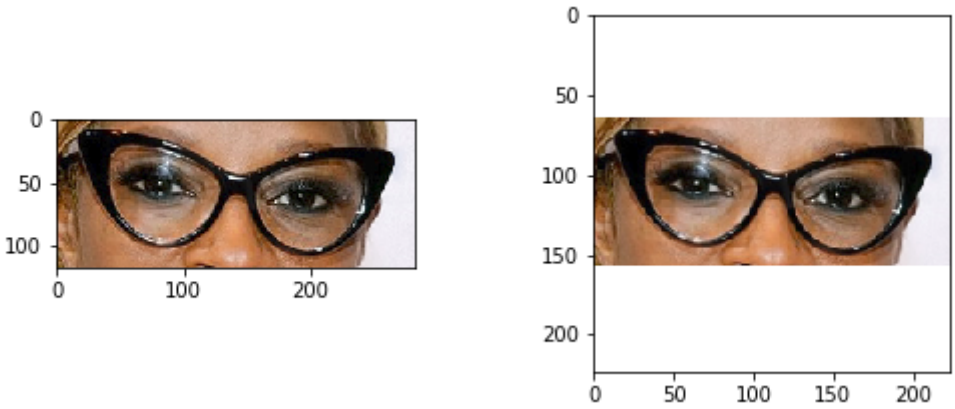


Abbildung 43: Aufbereitung der Bilder: Ausgangsbild, welches anhand der Bounding Box zugeschnitten wurde (links); für das Training aufbereitetes Bild im benötigten Format (rechts)

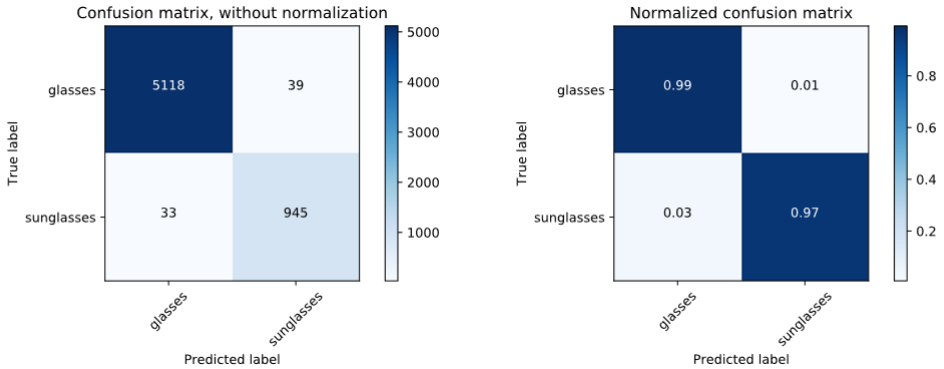


Abbildung 44: Klassifikationsmatrizen

Aufgrund der Simplizität der binären Klassifizierung konnte auf Mobilenet als Architektur zurückgegriffen werden, wobei die Imagenet-Gewichte geladen wurden. Der letzte Klassifika-tionslayer wurde entfernt und durch zwei Dense-Layer ersetzt, die die Kapazität des Netzes leicht erhöhten. Zudem wurde Dropout genutzt, um Overfitting zu vermeiden. Für den Output wurde entsprechend der binären Klassifizierung ein Dense-Layer mit zwei Neuronen verwendet. Das trainierte Modell erreichte eine Validation Accuracy von fast 99%, wie auch den Klassifikationsmatrizen zu entnehmen ist (siehe Abbildung 44).

Die wenigen falsch klassifizierten Bilder waren zum einen Teil Korrektionsbrillen, bei denen es durch eine Spiegelung im Glas den Anschein erweckte, als wären die Gläser getönt. Den anderen Teil machten Sonnenbrillen mit einer relativ hellen Verlaufstönung aus, die fälschlicherweise als Korrektionsbrillen erkannt wurden. Da der Großteil der Brillen jedoch korrekt klassifiziert, konnte diese Anforderung als erfolgreich abgeschlossen betrachtet werden.

SMT-ML.4 Klassifizierung der Brillenform Bei der Klassifizierung der Rahmenform handelte es sich erneut um ein Multiklassen-Problem. Entsprechend der Labelstrategie wurde zwischen Browline-Brillen, Cateye-Brillen, Brillen mit einer eckigen, hohen Form, Brillen mit einer eckigen, schmalen Form, ovalen Brillen, Pilotenbrillen, runden Brillen sowie sonstigen Brillen (z. B. Herzbrillen) unterschieden. Zudem wurde ein zusätzliches Label verwendet, mit welchem unbrauchbare Bilder gekennzeichnet wurden, bei denen die Object Detection die Brillen nicht richtig erkannte bzw. ein Objekt fälschlicherweise als Brille einstufte. Auf diese Weise sollte dem Modell für den praktischen Einsatz beigebracht werden, solche unbrauchbaren Bilder selbstständig zu erkennen, sodass dieses im weiteren Verlauf ignoriert werden konnten und für die Auswertungen in den Dashboards unbeachtet blieben.

Das Preprocessing konnte erneut von den zuvor beschriebenen Klassifizierungen übernommen werden, da die Labelinformationen auch hier wieder in einer einheitlichen CSV-Datei abgespeichert wurden. Bei der Modellarchitektur wurde sich ebenfalls an den vorherigen Problemstellungen orientiert und es kam ein auf Imagenet vortrainiertes MobileNet zum Einsatz. Es wurden auch andere Architekturen getestet, jedoch konnte keine nennenswerte Performancesteigerung erbracht werden. Die im Vergleich geringe Größe der MobileNet-Modelle war deshalb ausschlaggebend für die Entscheidung zugunsten eben dieser. Für das Training der Modelle wurden wie bei der Klassifizierung der Rahmenart Klassengewichte verwendet, da die 9 gelabelten Klassen im Datenset nicht mit der gleichen Häufigkeit vertreten waren.

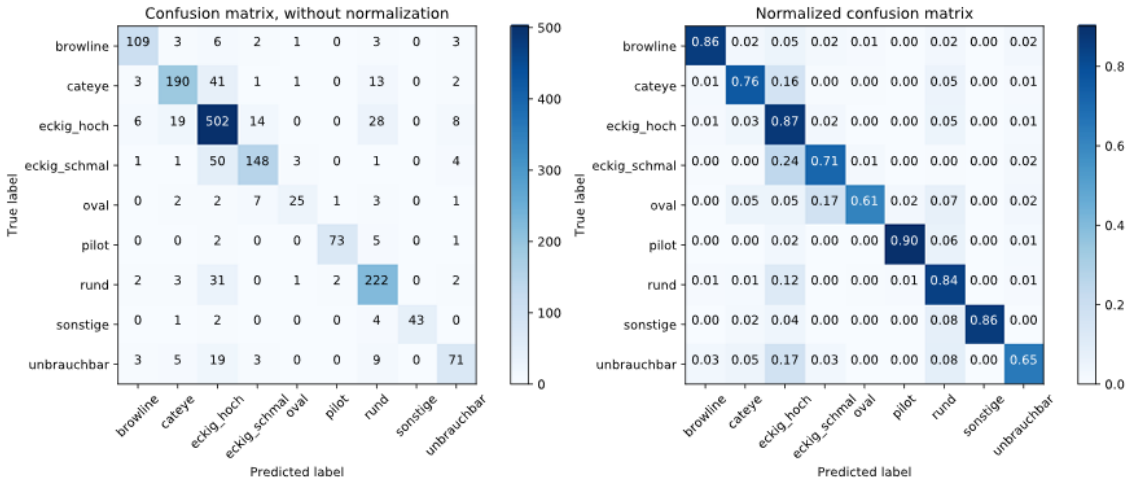


Abbildung 45: Klassifikationsmatrizen

Die Accuracy des finalen Modells betrug zirka 81%. Details hierzu können den Klassifikationsmatrizen in Abbildung 45 entnommen werden. In Anbetracht der hohen Anzahl an Klassen ist die Performance des Modells durchaus respektabel. Zum Vergleich: Bei einer zufälligen Klassifizierung der Rahmenform würde die Accuracy knapp 11% betragen. Überdies sind einige der Formen schwer voneinander zu unterscheiden, wie sich auch beim Labeln der Daten zeigte. Dies spiegelt sich in der Performance des Modells wider, was auch in der Klassifikationsmatrix erkannt werden kann. So zum Beispiel werden 24% der Brillen mit einer eckigen, schmalen Form fälschlicherweise als eckig, hohe Brillen klassifiziert. Der Übergang zwischen diesen beiden Brillenformen ist jedoch teilweise fließend, da die Formen in diesem Fall eher subjektiv sind und keine fest definierten Schwellwerte existieren, ab welcher Höhe eine Brille als hoch respektive schmal gilt.

Am schlechtesten performt das Modell bei den Klassen „oval“ und „unbrauchbar“. Im erstgenannten Fall kann dies darauf zurückgeführt werden, dass die Klasse im Datenset unterrepräsentiert ist. Überdies sind ovale Brillen teilweise schwer von runden sowie eckig schmalen Exemplaren zu unterscheiden, was sich auch in der Klassifikationsmatrix zeigt. Im zweitgenannten Fall, den unbrauchbaren Bilder, ist die Performance dadurch zu erklären, dass es sich ohnehin um eine Hilfsklasse handelt. Beim Labeln war es dem subjektiven Eindruck der entsprechenden Person überlassen, ab wann ein Bild als unbrauchbar gilt. Zudem weisen eben diese Bilder auch keine hohe Qualität auf, z. B. in Bezug auf die Auflösung. Dementsprechend war auch nicht zu erwarten, dass diese Klasse fehlerfrei erkannt wird. Vielmehr stellte es fast schon eine Überraschung dar, dass die Intention,

dem Modell beizubringen, unbrauchbare Bilder selbstständig zu erkennen, in die Tat umgesetzt werden konnte. Eine Accuracy von 65% ist auch hier wieder deutlich besser als der Zufall mit 11%. Zudem ist es durchaus möglich, dass das Modell in der Lage ist, Bilder zu klassifizieren, die für den Menschen nicht zweifelsfrei zu erkennen sind. Dies wäre auch eine mögliche Erklärung für die 35% der eigentlich unbrauchbaren Bilder, die vom Modell dennoch zu einer der Brillenformen zugeordnet wurden. Da das Modell in Bezug auf die anderen Klassen zuverlässig performte und auch die beiden negativen Ausreißer erklärt werden konnten, wurde die Anforderung insgesamt betrachtet erfolgreich abgeschlossen.

SMT-ML.5 Klassifizierung der Rahmenfarbe Für das Monitoring aktueller Trends ist die Beschreibung der Rahmenfarbe notwendig. Um diese Anforderung zu erfüllen, sollen die User-Bilder aus den selektierten Social-Media-Kanälen entsprechend klassifiziert werden. Die Einteilung der Kategorien orientiert sich an den Farben, die Brille24 als Attribute für ihre Produktbilder pflegt. Dementsprechend ergeben sich 12 Farben, beziehungsweise Farbanordnungen, als Zielsetzung für die Klassifizierung der Rahmenfarbe. Diese 12 Farben sind in Abschnitt 10.4 aufgelistet.

Diese Einteilung bringt den Vorteil mit sich, dass bereits eine Datenbasis besteht, die zum Training künstlicher neuronaler Netze verwendet werden kann. In diesem Dataset sind über 3000 Brillen hinterlegt, welche jeweils aus 5 Blickrichtungen fotografiert wurden, sodass insgesamt über 15000 Bilder mit entsprechenden Annotationen für das Training bereitstehen. Die Problematik bei der Verwendung der Produktbilder liegt nicht im Umfang des Datasets, sondern in der mangelnden Heterogenität. Die Bilder sind Studioaufnahmen mit schlichtem weißem Hintergrund und variieren demnach stark von den zu erwartenden User-Bildern aus den Social-Media-Kanälen.

Um diese Einschränkungen zu umgehen und das Dataset dennoch für die Klassifizierung nutzbar zu machen, wird vor der Klassifizierung eine semantische Segmentierung der Bilder durchgeführt.

Bei der semantischen Segmentierung wird das Bild in mehrere Bereiche aufgeteilt und diese ermittelten Bereiche werden einzelnen Objektklassen zugeordnet [RFB15]. In diesem Anwendungsfall wird zwischen Brillenrahmen und Hintergrund unterschieden, sodass im Anschluss der Segmentierung eine Maske erzeugt, die sich exakt mit dem Brillenrahmen überschneidet. Mittels dieser Maske wird dann der Brillenrahmen freigeschnitten und auf einem schlichten weißen Hintergrund platziert. Das erstellte Bild kann dann für die Klas-

sifizierung verwendet werden.

Für die Architektur der **Segmentierung** wird ein U-Net verwendet. Diese Netzarchitektur stellt einen Sonderfall eines Autoencoders dar und ist der State-of-the-Art für die Segmentier-ung von Bilddaten. Das ursprüngliche Einsatzgebiet, des an der Universität Freiburg entwickelten Ansatzes, liegt in der Biomedizin. Mit dem U-Net wurde eine Technik entwickelt, die es ermöglicht Zellmembranen in Mikroskop Aufnahmen pixelgenau zu kennzeichnen und dabei nur wenig Bilder für den Trainingsprozess benötigt.

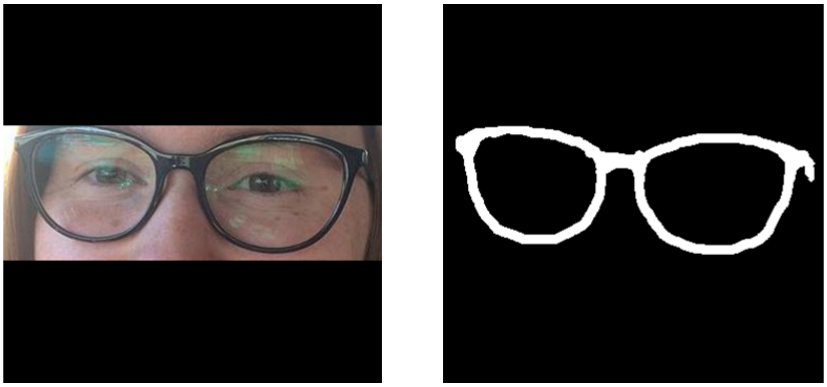


Abbildung 46: Beispielbild mit zugehöriger Maske aus dem erstellten Dataset

Neben der zu erwartenden Qualität der Segmentierung, ist der geringe Umfang an benötigten Trainingsdaten ein entscheidendes Argument für die Verwendung dieses Modell, da für die Entwicklung eines solchen Modells keine annotierten Daten zur Verfügung stehen. Dementsprechend muss ein eigenes Dataset aufgebaut werden. Um den späteren Prozess bestmöglich abzubilden, wird das vorhandene Instagram Dataset verwendet und die die Brillen mittels bestehenden Bounding Boxes ausgeschnitten. Diese Bildausschnitte werden dann zunächst mittels Padding in ein quadratisches Format transformiert und anschließend können Ground-Truth-Masken erzeugt werden. Diese Masken kennzeichnen den Bereich des Bildes, in dem sich der Brillenrahmen befindet. Die Brillengläser werden bei dem Annotieren bewusst nicht berücksichtigt, da in diesem Anwendungsfall nur die Farbe des Rahmens betrachtet werden soll. Für das Erstellen dieser Masken wird ein Bildbearbeitungs-programm verwendet (Gimp). Da das Annotieren der Masken sehr zeitaufwändig ist und für die Evaluation der Ergebnisse keine passende Metrik existiert, wird das Hard-Negativ-Mining angewandt. Dafür werden zufällig 500 Bilder aus dem Dataset ausgewählt. Im nächsten Schritt werden 50 Bilder, die untereinander eine möglichst große Varianz aufweisen, gesucht und annotiert. Anschließend wird das Netz mit diesen Bildern

trainiert und nach Abschluss des Trainings werden die Ergebnisse für alle 500 Bilder ausgewertet. Nun werden die 4 Fotos, auf denen das Netz die schlechtesten Ergebnisse erzielt, ausgewählt, annotiert und dann zum Trainingsset hinzugefügt. Nach erneutem Training erfolgt wieder eine Auswertung der Ergebnisse und dieser Prozess wird solange wiederholt bis die Ergebnisse für alle 500 Bilder ausreichend sind. Abschließend stehen für das letzte Training 94 Bilder mit Masken zur Verfügung. Ein entsprechendes Beispiel für die Annotation ist in Abbildung 46 dargestellt.

Die Architektur dieses Ansatzes besteht aus einem U-Net, welches mit einem vortrainierten Backbone ausgestattet ist. Dabei ist zu beachten, dass nur der Encoder vortrainiert ist. Als Input steht eine Batch an normalisierten und augmentierten Bildern bereit, welche in der Form 224*224*3 geladen werden. Der Output des Models muss in der Höhe und in der Breite die gleichen Maße wie der Input aufweisen. Da für jeden Pixel nur eine binäre Klassifizierung erfolgt, liegt der Output in der Form 224*224 vor.

Die Besonderheit der Trainingsroutine bezüglich der Datenbeschaffung wurde bereits erläutert. Für das eigentliche Training des Modells wird die „binary cross entropy“ verwendet, um den Loss der Vorhersagen zu ermitteln und als Learning Rate wird 0,001 festgelegt. Bei jedem neuen Trainingsdurchgang werden die Gewichte des Encoders für die ersten zwei Epochen eingefroren, damit der Effekt des Transfer Learnings optimal ausgenutzt wird. Beim finalen Training mit den 94 annotierten Bildern stagniert der Loss nach 260 Epochen und das Modell ist ausreichend trainiert.

Abbildung 47: Anwendung der Segmentierung

Nachdem das Modell der Segmentierung entwickelt wurde, kann es jetzt in der in der Pipeline implementiert werden, um den Input für die Klassifizierung der Rahmenfarbe zu erzeugen. Dafür segmentiert das Modell das entsprechende Foto, welches der Crawler liefert. Mit der entstandenen Maske werden dann die Bereiche des Bildes ausgeschnitten, in dem sich der Brillenrahmen befindet. Ein Beispiel für diesen Prozess ist in Abbildung 47 dargestellt. In diesem Beispiel wird das Bild aufgegriffen, welches die Objekt Detection in Abbildung 42 zurückgegeben hat.

Nachdem die Segmentierung der des Rahmens abgeschlossen ist, kann mit der **Klassifizierung der Rahmenfarbe** begonnen werden. Das Klassifizierungsproblem unterscheidet sich dabei von den bisherigen Aufgaben dieses Projektes. In der Regel wurden Klassifizierungen so formuliert, dass es Menge an Klassen gab aus der eine ausgewählt werden musste. In Abschnitt 10.4 wurden bereits die 12 Farbkategorien vorgestellt, in welche die Bilder eingeordnet werden sollen. In diesem Fall ist es allerdings möglich, dass eine Brille mehreren Klassen zugeordnet wird. Beispielsweise kann eine Brille sowohl meliert als auch braun sein. Dementsprechend ist ein one-hot-encoding nicht mehr möglich und es muss ein k-hot-encoding gewählt werden.

Die Datengrundlage wird für die gewählte Herangehensweise in zwei Bereiche aufgeteilt. Der erste Teil besteht aus den Produktbilder, die von Brille 24 bereitgestellt werden. Dabei werden die Bilder ausgeschlossen, für die keine Beschreibung der Rahmenfarbe existiert. Daneben werden 1500 Bilder selbst annotiert, die aus dem Instagram Dataset hervorgehen. Dies ist erforderlich, da die Produktbilder den späteren Anwendungsfall nur begrenzt abbilden können. Im Rahmen des Preprocessings werden die normalisierten und augmentierten Bilder in einer Batch geladen und stehen abschließend in der Form 224*224*3 für das Training bereit. Der Output ist ein Numpy-Array mit 12 Elementen, von denen jedes eine Farbe binär kodiert.

Die gewählte Architektur ist ein MobileNetV2, welches vortrainierte Gewichte verwendet. Die Wahl fiel auf dieses Modell, da das Klassifizierungsproblem verhältnismäßig einfach ist und bereits bei der Segmentierung eine rechenintensive Architektur verwendet wird. Eine Besonderheit beim Training dieses Modells besteht darin, dass „binary cross entropy“ als Loss-Funktion gewählt wird, obwohl es sich um eine Klassifizierung mit mehreren Klassen handelt, bei der gewöhnlich die „categorical cross entropy“ verwendet wird. Dadurch wird der Loss dem Umstand gerecht, dass es sich um ein k-hot-encoding und nicht um ein one-hot-encoding handelt. Das eigentliche Training des Modells ist in drei Abschnit-

te unterteilt. Im ersten Schritt wird das Dataset mit den Produktbildern von Brille24 verwendet. Der Trainingsprozess beginnt damit, dass die ersten Convolutional Layer des Modells eingefroren werden und dieses dann für 10 Epochen trainiert wird. Dadurch wird wieder sichergestellt, dass die Effekte des Transfer Learnings optimal ausgenutzt werden. Im nächsten Schritt werden alle Schichten für das Training freigegeben und das Netz wird bis zum optimalen Zustand trainiert. Nach diesem Schritt existiert ein Modell, welches auf den Produktbilder gute Ergebnisse erzielt, aber beispielsweise bei unsauber segmentierten Bildern unsicher reagiert. Um diesen Umstand auszugleichen wird im letzten Schritt das selbst annotierte Dataset verwendet. Auf Basis dieser Datengrundlage wird das Modell mit einer um den Faktor 10 reduzierten Learning Rate für weitere 20 Epochen trainiert.

Abschließend kann ein Verfahren in die Pipeline integriert werden, welches die gecroppten Fotos zunächst segmentiert und diese Bilder anschließend entsprechend der 12 Farbmerkmale klassifiziert. Damit ist diese Anforderung erfolgreich umgesetzt.

SMT-ML.6 Klassifizierung der Brillenart Bei der Klassifizierung der Rahmenart handelt es sich um ein Multiklassen-Problem, wobei zwischen Vollrand- und Halbrandbrillen sowie rahmenlosen Modellen unterschieden werden soll. Da die gleiche Datengrundlage wie für die Differenzierung von Korrektions- und Sonnenbrille verwendet wurde, war das Preprocessing relativ simplen Charakters, lediglich die CSV-Datei mit den Labelinformationen musste ausgetauscht werden. Als Architektur wurde erneut auf Mobilenet zurückgegriffen, wobei der letzte Denselayer entsprechend der Problemstellung aus drei Neuronen bestand. Beim Training gab es zwei wesentliche Neuerungen. Zum einen wurde Data Augmentation verwendet, um Overfitting zu vermeiden, zum anderen wurden Klassengewichte verwendet, um der Tatsache, dass das Datenset nicht ausbalanciert war, gerecht zu werden. So traten Vollrandbrillen mit Abstand am häufigsten auf, gefolgt von Halbrandbrillen, wohingegen randlose Brillen eher eine Ausnahmeerscheinung im Datenset darstellten. Das erste trainierte Modell wies trotz dieser Anpassungen eine eklatante Schwäche auf, nämlich dass es in der Regel nicht dazu in der Lage war, Halbrandbrillen zu erkennen. Es stellte sich heraus, dass die Bilder nicht einheitlich gelabelt wurden, Brillen mit einem sehr dünnen unteren Rahmen wurden von manchen Personen als Halbrandbrille eingestuft, andere Labelfachkräfte hingegen ordneten sie der Klasse der Vollrandbrillen zu. Aufgrund dieser Inkonsistenz in den Daten, mussten diese erneut gelabelt werden. Ein besonderes Augenmerk lag dabei auf besagten Brillen mit einem dünnen unteren Rahmen. Teilweise war dies allerdings sehr schwierig zu erkennen, da nicht ausgemacht werden konnte, ob es sich um einen Rahmen handelt oder ob sich das Licht im Glas spiegelt.

Abbildung 48: Eine kleine Auswahl der beschriebenen Härtefälle

Zudem wurde bei einer genaueren Untersuchung des Datensets deutlich, dass die Object Detection in einigen Fällen die Brillen nicht richtig erkannt hatte oder die zugeschnittenen Bilder schlichtweg eine zu geringe Auflösung hatten, was es unmöglich machte, das genaue Label zu bestimmen. Die betroffenen Bilder wurden dementsprechend aussortiert. Mithilfe der angepassten Datengrundlage konnte ein neues Modell trainiert werden, welches deutlich besser performte. Details können auch hier der Klassifikationsmatrix entnommen werden (siehe Abbildung 49).

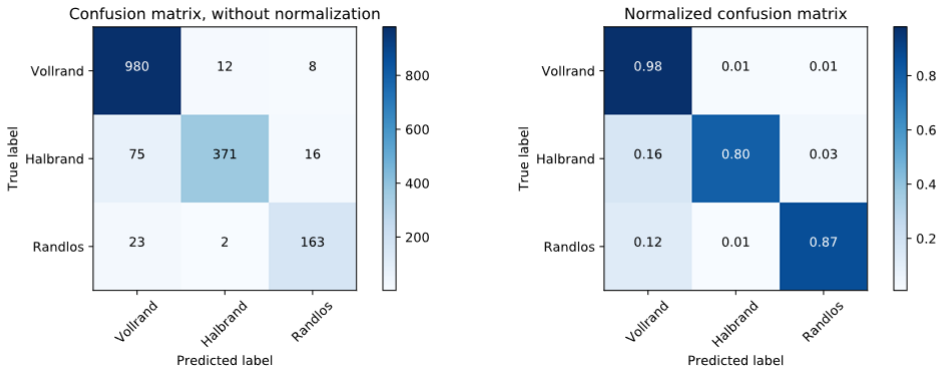


Abbildung 49: Klassifikationsmatrizen

Wie zu erkennen ist, werden Vollrandbrillen vom Modell zuverlässig erkannt. Halbrandbrillen werden in 16% der Fälle fälschlicherweise als Vollbrandbrille klassifiziert, was auf die zuvor erläuterte Problematik zurückzuführen ist. Diesbezüglich erwies es sich jedoch selbst für den Menschen teilweise als schwierig, zwischen den beiden Klassen zu unterscheiden, weswegen die Performance des Modells durchaus respektabel ist. Die Tatsache, dass randlose Brillen teilweise falsch klassifiziert werden, lässt sich wohl darauf zurückführen, dass diese Klasse unterrepräsentiert ist, eine korrekte Einstufung in 87% der Fälle ist dennoch positiv hervorzuheben. Die Anforderung wurde somit insgesamt erfolgreich abgeschlossen.

10.5.5 Abnahme

Folgend wird die Erfüllung der einzelnen Anforderungen bewertet. Erfüllte Anforderungen werden mit **Abgenommen**, nicht erfüllte Anforderungen mit **nicht Abgenommen** markiert. Anforderungen, die auf andere Art und Weise gelöst wurden erhalten die Markierung **nicht umgesetzt**.

Nr.	Beschreibung	Abgenommen
SMT-ML.1	Erkennung einer Brille auf Bild	**Abgenommen**
SMT-ML.2	Klassifizierung des Geschlechts einer zu erkennenden Person	**Abgenommen**
SMT-ML.3	Klassifizierung Brille oder Sonnenbrille	**Abgenommen**
SMT-ML.4	Klassifizierung der Brillenform	**Abgenommen**
SMT-ML.5	Klassifizierung der Rahmenfarbe	**Abgenommen**
SMT-ML.6	Klassifizierung der Brillenart	**Abgenommen**

Tabelle 24: Abnahme der Anforderungen an das Machine Learning

10.6 Fazit

Die Trendanalyse hat viele Komponenten die zusammenspielen müssen, um einen reibungslosen Ablauf zu gewährleisten. Hierbei konnten von allen Komponenten auch alle Anforderung erfüllt werden. Das Dashboard bietet auf Grundlage der gesammelten Bilder des Crawlers und der zusätzlichen analysierten Informationen durch das Machine Learning die Möglichkeit, verschiedene gegenwärtige und historische Trends zu erkennen. Genau hier liegt aber auch ein Problem, denn folglich wird nur ein IST-Zustand erörtert. Um einen zukünftigen Trend erkennen zu können, müsste das Dashboard mit Prognosen erweitert werden. Zudem ist der Crawler sehr wartungsintensiv, da er auf Klassennamen oder IDs verschiedener Webelemente zugreift, um entsprechende Informationen zu sammeln. Da diese aber häufig von Webseiten geändert werden, müssen im Crawler entsprechende Anpassungen stattfinden, damit es zu keinen Abbrüchen oder fehlenden Informationen kommt.

10.7 Ausblick

Wie bereits im Fazit angedeutet, wäre es wichtig, Prognoseverfahren für eine tatsächliche Trenderkennung zu erarbeiten und zu integrieren, da die jetzige Umsetzung eher den IST-Zustand des Marktes beschreibt und somit nur aktuelle Trends erkennbar sind. Da die Trendanalyse allerdings die Unterstützung des Einkaufs in Vordergrund sieht und dieser

meistens die Trends vorausschauend erkennen muss, da die Brillen für einen Saison meistens bereits ein halbes Jahr früher gekauft werden, sehen wir hier die wichtigste Baustelle, an der angesetzt werden könnte.

Des Weiteren könnten weitere Modelle klassifiziert werden, wie beispielsweise die Tönung von Sonnenbrillen, damit weitere Informationen generiert und somit auch weitere und eventuell unbekannte Trends abgleitet werden können.

Zudem hat Brille24 ein neues Projekt mit dem Titel Sichtwechsel, bei dem es um eine Street2Shop Anwendung geht. Dabei kann der Nutzer eine Brille im Alltag fotografieren und entsprechend aussehende Brillen auf der Webseite vorgeschlagen bekommen. Die dadurch gesammelten Daten könnten mit in die Analyse einbezogen werden, da hierdurch weitere Trends aufgezeigt werden können, die noch näher an den Bedürfnissen und Wünschen der eigentlichen Kunden von Brille24 angelehnt sind.

11 Brillenpassleser

Mit dem Brillenpassleser wird das Ziel verfolgt dem Kunden den Bestellprozess zu erleichtern, indem die Sehwerte durch das Abfotografieren seines Brillenpasses mittels Bild- und Texterkennung automatisch erfasst werden.

11.1 Beschreibung

Viele Optiker, wie bspw. der Marktführer Fielmann, händigen zu jeder Brille einen sogenannten Brillenpass aus, auf dem die folgenden Informationen vermerkt werden:

- Die Sehwerte
- Die Pupillendistanz
- Die Anschrift der Filiale
- Der Name des Kunden
- Das verwendete Glas
- Die Fassung

Für die Online-Bestellung einer Korrektionsbrille sind die Sehwerte sowie die Pupillendistanz erforderlich. Diese müssen daher von dem Brillenpassleser ausgelesen werden können. Optional kann auch der Name des Kunden auf dem Brillenpass mit den Daten des Bestellers abgeglichen werden, um diesen auf die eventuelle Verwendung eines falschen Brillenpasses hinzuweisen. Um die Nutzung des Brillenpasslesers so einfach wie möglich zu gestalten, soll dieser mittels einer Webapp umgesetzt werden, damit keine dedizierte App installiert werden muss. Des Weiteren sollen die Werte, nachdem ein Brillenpass eingelesen wurde, dem Benutzer angezeigt und durch diesen bestätigt werden. Dieser Schritt ist notwendig, um Einlesefehler berichtigen zu können.

11.2 Anforderungen

Die gesammelten Anforderungen werden in insgesamt zwei Kategorien eingeteilt. Die erste Kategorie betrifft die Anforderungen an die Benutzeroberfläche und die zweite Kategorie das Servicelayer. Alle Anforderungen wurden in Zusammenarbeit mit dem Praxispartner erstellt.

11.2.1 Benutzeroberfläche/Webapp

Die Benutzeroberfläche bildet die Schnittstelle zum Benutzer. Hier wird der Brillenpass abfotografiert und die ausgelesenen Daten bestätigt bzw. geändert. Die gesammelten Anforderungen der Benutzeroberfläche sind in Tabelle 25 zusammengefasst.

Nr.	Beschreibung
BPL.1.1	**Aufnahmen eines Fotos mit der Gerätekamera** Der Nutzer kann mit seinem mobilen Endgerät ein Foto des Brillenpasses aufnehmen. Zur Unterstützung des Nutzers wird ein Rahmen eingeblendet, in dem sich der Brillenpass befinden muss.
BPL.1.2	**Verifizieren des aufgenommen Fotos durch den Nutzer** Nach der Aufnahme wird der Nutzer dazu aufgefordert das Foto zu prüfen und anschließend zu bestätigen. Die Bestätigung soll mittels eines Buttons erfolgen, außerdem soll es die Möglichkeit geben den Vorgang abzubrechen und ein neues Foto aufzunehmen.
BPL.1.3	**Senden des Fotos an den HTTP-Endpunkt des Servicelayers** Das aufgenommene Foto kann an den HTTP-Endpunkt des Servicelayers gesendet werden. Die Antwort enthält die ausgelesenen Werte.
BPL.1.4	**Anzeigen der ausgelesenen Werte** Strukturierte und gut sichtbare Anzeige der ausgelesenen Werte.
BPL.1.5	**Ändern der ausgelesenen Werte** Der Nutzer kann Werte die nicht oder fehlerhaft erkannt wurden nachtragen bzw. korrigieren.
BPL.1.6	**Validieren der ausgelesenen Werte** Die ausgelesenen Werte werden durch den Nutzer mit einem Button bestätigt oder der Vorgang wird abgebrochen.
BPL.1.7	**Rückgabe der Werte an einen beliebigen HTTP-Endpunkt** Die gesammelten Daten können ab einen beliebigen Endpunkt übertragen werden (Verbindung zum Webshop).

Tabelle 25: Anforderungen an die Benutzeroberfläche

11.2.2 Servicelayer

Das Servicelayer bildet innerhalb der Architektur die eigentlichen Services an. Hier werden z. B. Dienste wie die Schrifterkennung angeboten. Die gesammelten Anforderungen des Servicelayers sind in Tabelle 26 zusammengefasst.

Nr.	Beschreibung
BPL.2.1	**Empfangen von Fotos** Der Servicelayer muss in der Lage sein Anfragen über HTTP zu erhalten und diese zu beantworten.
BPL.2.2	**Finden und ausschneiden des Brillenpasses** Der Brillenpass soll auf dem empfangenen Bild identifiziert und ausgeschnitten werden.
BPL.2.3	**Klassifizierung des Herausgebers** Der Herausgeber des Brillenpasses soll klassifiziert werden, damit die Zuordnung von Text zu den entsprechenden Sehwerten erleichtert wird.
BPL.2.4	**Erkennung der gedruckten Werte** Die Werte sollen identifiziert und als Text gespeichert werden. Außerdem soll die Position im Bild, an der der Text gefunden wurde, gespeichert werden.
BPL.2.5	**Zuordnen der ausgelesenen Werte zu Eigenschaften (Sphere, Addition usw.)** Die ausgelesenen Werte können den verschiedenen Eigenschaften aufgrund ihrer Position auf dem Brillenpass zugeordnet werden.
BPL.2.6	**Rückgabe der ausgelesenen Werte in einem standardisierten Datenformat** Die Daten müssen in einem Datenformat zurückgegeben werden, welches vom Frontend verarbeitet werden kann und in jedem Fall gleich ist.
BPL.2.7	**Erzeugung von Testdaten** Zum Test der Ergebnisse wird ein Werkzeug zur Erzeugung von Testdaten benötigt. Dieses muss mindestens in der Lage sein Brillenpässe der Anbieter Brille24 und Fielmann zu erzeugen.

Tabelle 26: Anforderungen an den Servicelayer

11.3 Umsetzung

Im Folgenden wird die Umsetzung des Brillenpasslesers vorgestellt. Die Beschreibung der Implementierung orientiert sich dabei an der, in den Anforderungen, definierten Gliederung. Dementsprechend wird zunächst die Umsetzung der Webapp vorgestellt, bevor dann die Umsetzung des Servicelayers folgen.

11.3.1 Webapp

Das Frontend des Brillenpasslesers basiert auf den TypeScript-Framework Angular. Um eine einfache Einbindung in vorhandene Systeme zu gewährleisten soll allerdings möglichst auf die Verwendung von Angular-Komponenten verzichtet und stattdessen auf reines JavaScript gesetzt werden. Als Basis dient allerdings dennoch Angular, da hier eine Vielzahl

von Designelementen in Form von CSS-Klassen enthalten sind, die zur Erstellung der Oberfläche verwendet werden können. Die Umsetzung der Webapp wird anhand der definierten Anforderungen vorgestellt.

BPL.1.1 Aufnahme eines Fotos mit der Gerätekamera Um die Daten des Brillenpasses auszulesen wird ein Foto von diesem benötigt. In den Anforderungen wurde definiert, dass zum aufnehmen des Fotos die Gerätekamera genutzt werden soll. Alternativ wäre auch ein Upload per Formular möglich, diese Möglichkeit wurde allerdings nicht implementiert, da angenommen wurde, dass kaum Anwender ein Foto ihres Brillenpasses auf ihrem Gerät speichern.
Der Zugriff erfolgt auf allen Plattformen und in allen Browsern über die JavaScript-Methode *mediaDevices.getUserMedia()*. Dieser werden Bedingungen (Constraints) übergeben die z. B. festlegen, welche Kamera genutzt werden soll (Front- oder Hauptkamera), welche Auflösung gefordert wird und ob das Mikrofon des Gerätes ebenfalls angefordert werden soll. Die Funktion zum Aufruf der Kamera *initCamera()* wird in der folgenden Abbildung dargestellt.

```
1  this.config = { audio: false, video: { width: 1920, facingMode: { exact: "
       environment" } }};
2
3  initCamera() {
4    const browser = <any>navigator;
5
6    browser.mediaDevices.getUserMedia(this.config).then(mediaStream => {
7    this.mediaStream = mediaStream;
8    this.video.srcObject = this.mediaStream;
9    this.video.play()
10     .then(() => this.createVideoOverlay());
11   }).catch(error => this.handleCameraError(error));
12 }
```

Listing 13: Zugriff auf Gerätekamera

Im ersten Schritt werden die Constraints definiert (Bildbreite 1920px, Auswahl der Hauptkamera). Mit diesen Constraints wird anschießend die Kamera angefragt und das zurückgelieferte Video (*mediaStream*) als Quellobjekt eines Video-HTMLElements gesetzt. Anschließend wird das Video mit der Funktion *play()* gestartet. Schlägt die Anforderung fehl wird die Funktion *handleCameraError(error)* aufgerufen, um den Fehler zu behandeln. Diese prüft, ob es sich um einen Fehler vom Typ OverconstrainedError (Constraints können vom Gerät nicht erfüllt werden) handelt. Trifft dies zu werden neue Constraints gesetzt und die Ka-

mera erneut angefordert. In jedem anderen Fall wird die Methode *handleError(error)* aufgerufen, die den Fehler in der JavaScript-Console ausgibt und dem Anwender einen Fehler anzeigt.

```
handleCameraError(error){
  if(error.name ==='OverconstrainedError'){
    this.config = { audio: false, video: { width: 1280 } };
    this.initCamera();
  }else{
    this.handleError(error);
  }
}
```

Listing 14: ErrorConstraint für Gerätekamera

Tritt kein Fehler auf wird im letzten Schritt ein Rahmen über das Video gezeichnet, welcher dem Nutzer helfen soll ein korrektes Foto des Brillenpasses aufzunehmen. Anschließend kann der Nutzer per Klick auf einen Auslösebutton ein Foto aufnehmen. Das Foto wird anschließend in einen HTML-Canvas gezeichnet und der Zugriff auf die Gerätekamera beendet.

BPL.1.2 Verifizieren des aufgenommen Fotos durch den Nutzer Um sicherzustellen, dass der gesamte Brillenpass sichtbar ist und das Foto der benötigten Qualität entspricht soll das aufgenommene Foto vom Nutzer überprüft und freigegeben werden. Hierzu werden unterhalb des Bildes zwei Button angezeigt. Mit einem Button wird der Vorgang abgebrochen und der Nutzer gelangt zurück zur Kamera. Mit dem Zweiten Button wird das Foto freigegeben und an das Servicelayer gesendet.
Um den Brillenpass auf dem Bild zu finden sollte ursprünglich Machine Learning in Form eines neuronalen Netzes eingesetzt werden. Dies Ansatz wurde allerdings aus zeitlichen Gründen zunächst verworfen. Stattdessen wurde die Aufgabe an den Nutzer übertragen, welcher die Ecken des Brillenpasses markiert, wobei die Punkte nach dem Setzen per Drag-and-Drop verschoben werden können. Realisiert wurde die Funktion mittels Eventlistener in JavaScript. Da das Foto auf mobilen Endgeräten nur mit 80% den vollständigen Größe angezeigt werden kann müssen die Pixelwerte auf Werte zwischen 0 und 1 normalisiert werden, damit nach der Übertragung an das Servicelayer die Position auf dem original großen Bild berechnet werden kann. Die normalisierten Koordinaten werden abschließend in einer Liste gespeichert. Im Laufe des Projektes wurde diese Funktion wieder entfernt und doch durch ein Machine Learning Verfahren ersetzt (siehe BPL.2.2).

BPL.1.3 Senden des Fotos an den HTTP-Endpunkt des Servicelayers Das Frontend des Brillenpasslesers wird aus logischer Sicher außerhalb des Servicelayers betrieben. Aus diesem Grund erfolgt die Kommunikation mit dem im Servicelayer betriebenen Backenddienst über das Servicegateway (siehe Kapitel 7), die Anfrage muss demnach an den HTTP-Endpunkt des Gateway gesendet werden. Hierzu wird der in Angular integrierte HTTP-Client verwendet. Dieser serialisiert das in den vorherigen Schritten erstellte Objekt und sendet dieses als Request an das Servicegateway. Als Antwort erhält der Client die ausgelesenen Werte.

BPL.1.4-6 Anzeigen, Ändern und Validieren der ausgelesenen Werte Wie bereits beschrieben, werden die Ergebnisse der Texterkennung im Servicelayer als Antwort zurück an die Webapp gesendet. Hier werden die ausgelesenen Werte in einem Formular angezeigt. Hierbei werden die ausgelesenen Werte in einem Inputfeld angezeigt. Konnte ein Wert nicht ausgelesen werden, bleibt das Feld leer. Diese Darstellungsform ermöglicht es dem Nutzer die Werte zu ändern bzw. nicht ausgelesene Daten hinzuzufügen und anschließend zu versenden. Über einen Button unterhalb des Formulars bestätigt der Nutzer die Korrektheit der Werte.

BPL.1.7 Rückgabe der Werte an einen beliebigen HTTP-Endpunkt Nach der Validierung der Werte müssen diese an das Brille24-Shopsystem übertragen werden. Hierzu wird ebenfalls der Angular HTTP-Client verwendet. Dieser schickt die Werte in einem standardisierten JSON-Schema.

11.3.2 Servicelayer

Das Servicelayer stellt den eigentlichen Hauptbestandteil des Brillenpasslesers dar. Hier werden die vom Frontend ankommenden Bilder verarbeitet. Folgend wird die Umsetzung der einzelnen Anforderungen beschrieben.

BPL.2.1 Empfangen von Fotos Um Fotos zu empfangen wurde im ersten Schritt ein HTTP-Endpunkt mit dem Phyton Webframework Flask implementiert. Dieser stellte einen HTTP-Endpunkt bereit, welcher beliebigen Inhalt empfangen kann. Aufgrund des einfachen Aufbaus eignet sich das Framework sehr gut, um schnell einen nutzbaren Service zum Testen aufzubauen.
Im späteren Verlauf wurde der Endpunkt durch die in Kapitel 5 beschriebene Basisbibliothek ersetzt.

BPL.2.2 Finden und ausschneiden des Brillenpasses Um die Werte des Brillenpasses auslesen zu können muss dieser im ersten Schritt auf den aufgenommenen Bild identifiziert und ausgeschnitten werden. Hierzu wurden insgesamt drei möglich Wege identifiziert:

- Erkennung mittels Canny-Edge-Detection im Servicelayer
- Erkennung und Markierung durch den Benutzer
- Erkennung mittels KI im Servicelayer

Der Grundgedanke ist bei jeder der drei Herangehensweisen identisch. Sowohl für das Auslesen des Textes als auch für die spätere Zuordnung zu den entsprechenden Sehwerten ist es notwendig, dass die fotografierten Brillenpässe immer im gleichen Blickwinkel abge-bildet sind. In diesem Anwendungsfall sollte der Brillenpass exakt senkrecht aus der Vogelperspektive und ohne jegliche Neigung aufgenommen sein. In der Praxis kann nicht erwartet werden, dass die Nutzer eine entsprechende Aufnahme bereitstellen. Dementspre-chend muss eine Möglichkeit gefunden werden, welche die Aufnahmen des Kunden in ein entsprechendes Format überführt. OpenCV stellt eine Möglichkeit bereit, um transformierte Aufnahmen von Objekten zu entzerren. Allerdings benötigt es dafür die Eckpunkte des zu bearbeitenden Objektes.

Zu Beginn des Projektes wurden die Eckpunkte des fotografierten Brillenpasses manuell annotiert, in dem der Nutzer gebeten wurde, nach Aufnahme des Bildes, die Eckpunkte des entsprechenden Brillenpasses zu markieren. Die Entscheidung fiel zugunsten dieser Herangehensweise, weil der Fokus zunächst auf der Selektion der Textfelder lag. Im Verlauf des Projektes konnten, auch durch die Arbeit an anderen Teilprojekten, weitere Kompeten-zen erworben werden. Während der Arbeit an der Landmark Detection bei dem Teilprojekt Brillenberater wurden Verfahren entwickelt, welche ebenfalls beim Brillenpassleser eingesetzt werden können.

Als Folge dieser Entwicklung wird ein CornerDetector nach dem Yolo-Prinzip entworfen. Netzarchitektur und Trainingsroutine können dem entsprechenden Abschnitt aus dem Brillenberater entnommen werden und werden an dieser Stelle unverändert beim Passleser angewandt. Unterschiede bestehen nur hinsichtlich der Datengrundlage und des Outputs.

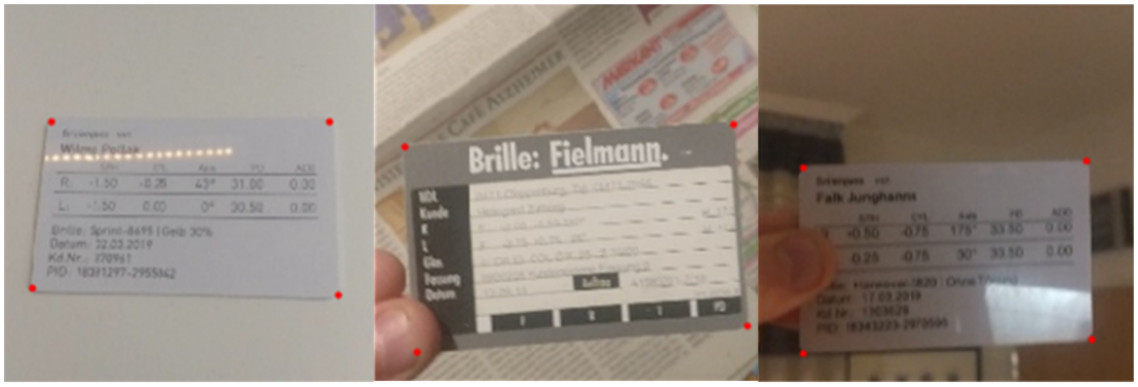


Abbildung 50: Trainingsdaten CornerDetector - Beispiele [Eigene Darstellung]

Datengrundlage: Für das Training des CornerDetectors stehen 2491 Fotos von Visitenkarten zur Verfügung. Das Dataset wurde von Brille24 Research bereitgestellt und enthält neben den Bildern auch eine Liste mit den zugehörigen Eckpunkten der Visitenkarten. Obwohl diese Visitenkarten den Brillenpässen hinsichtlich Form und Struktur stark ähneln, werden dem Dataset zusätzlich 112 Fotos von Brillenpässen hinzugefügt. Bei der Auswahl der zu annotierenden Bilder liegt ein besonderes Augenmerk darauf, dass die Trainingsdaten viele der zu erwartende Szenarien – wie bspw. Lichtspieglungen oder komplexe Hintergründe – abbilden. Drei Beispiele für diese erstellten Trainingsdaten sind in Abbildung 50 dargestellt.

Trainingsoutput: Gegenüber dem Yolo-Ansatz bei der Facial Landmark Detection, muss neben der Datengrundlage auch der Output von Trainings- und Testdaten überarbeitet werden. Bei dem Brillenberater wurde das Regressionsproblem auf die 11 wesentlichen Landmarks reduziert, damit der Yolo-Ansatz eingesetzt werden kann. In diesem Fall werden die vier Eckpunkte eines Brillenpasses als Objekte definiert und der Output ist hier dementsprechend schlanker.

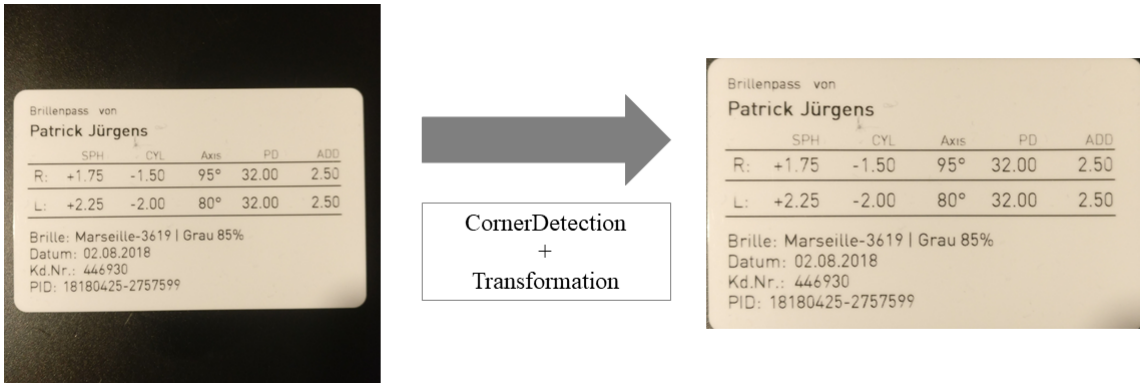


Abbildung 51: Anwendung des CornerDetectors [Eigene Darstellung]

In Abbildung 51 ist die Implementierung des CornerDetectors dargestellt. Aus dem Foto des Kunden wird der mittlere Bereich quadratisch ausgeschnitten. Dies ist möglich, da dem Kunden durch eine Schablone verdeutlicht wird, dass der Brillenpass in diesem Bereich platziert werden soll. Nach der Ermittlung der vier Eckpunkte, werden diese mit dem entsprechenden Foto an eine Funktion übergeben, welche den Brillenpass transformiert, falls der Pass auf dem Bild verzerrt dargestellt ist, und anschließend ausschneidet. Bei der Implementierung des Yolo-Ansatzes im Brillenberater (vgl. Gesichtsform Brillenberater) trat die Problematik auf, dass unter Umständen nicht alle definierten Landmarks gefunden werden können. Dieser Fall kann auch bei der vorliegenden Implementierung eintreffen. Dies ist meistens der Fall, wenn ein Anwender Fehler bei der Aufnahme gemacht hat und der Brillenpass beispielsweise nicht vollständig im Aufnahmebereich ist. Sobald weniger als vier Eckpunkte erkannt werden, erhält der Nutzer eine Fehlermeldung mit einem erneuten Verweis auf die Aufnahmehinweise. Da für jede Rasterzelle eigene Vorhersagen getroffen werden, kann auch der Fall eintreten, dass mehrere Vorhersagen für ein Objekt über dem Schwellenwert liegen (z. B. existiert zweimal der obere rechte Eckpunkt). In diesem Fall wird nur das Ergebnis mit dem höchsten Prediction Score für ein Objekt zurückgegeben. Dadurch kann sichergestellt werden, dass jedes Objekt der Klassifizierung nur einmal erhalten bleibt und dementsprechend maximal vier Eckpunkte zurückgegeben werden.

Durch die zweite Implentierung, der Erkennung mittels KI im Servicelayer, kann der Brillenpass jetzt eigenständig erkannt und ausgeschnitten werden. Die Anforderung ist somit erfüllt.

BPL.2.3 Klassifizierung des Herausgebers Auf den Brillenpässen der verschiedenen Anbieter stehen die auszulesenden Werte an unterschiedlichen Positionen. Daher muss erkannt werden, um welchen Brillenpass es sich handelt. Dies wurde zunächst durch einen Histogrammvergleicher realisiert. Dazu werden zwei Beispielbilder von den Brillenpässen eingefügt, aus denen Histogramme als Referenz generiert werden. Wird nun ein Bild geladen, wird von diesem ein Histogramm erstellt. Dieses wird mit den beiden Histogrammen der Referenzbilder verglichen. Das Bild vom Brillenpass wird dem Anbieter zugeordnet, bei dem das Histogramm am ähnlichsten ist. [Daw14]

Ein zentrales Problem beim Histogrammvergleicher stellt die unterschiedliche Belichtung der Bilder dar. Bei zu dunkler oder zu heller Belichtung kann es zu einer falschen Klassifikation kommen. Deswegen wurde die Entscheidung getroffen, ein anderes Verfahren auszuprobieren.

Dabei identifiziert ein Feature Matching Ähnlichkeiten in der Bildstruktur. Dieser Feature Matcher ist Teil der OpenCV Bibliothek und geht nach dem BruteForce Verfahren vor. Aus dem ersten Bild wird ein Merkmal mit allen Merkmalen aus dem zweiten Bild verglichen. Dann wird der Punkt mit der geringsten Distanz, also der größten Ähnlichkeit gespeichert. Anschließend wird der nächste Punkt aus dem ersten Bild mit allen aus dem zweiten Bild verglichen. Dies wird für alle Punkte durchgeführt [MM12].

Um nun zwischen den Brillenpässen zu differenzieren, wird die Summe aus den 20 Punkten mit der geringsten Distanz ermittelt. Das Referenzbild bei dem die Summe geringer ist, kann somit dem passenden Anbieter zugeordnet werden.

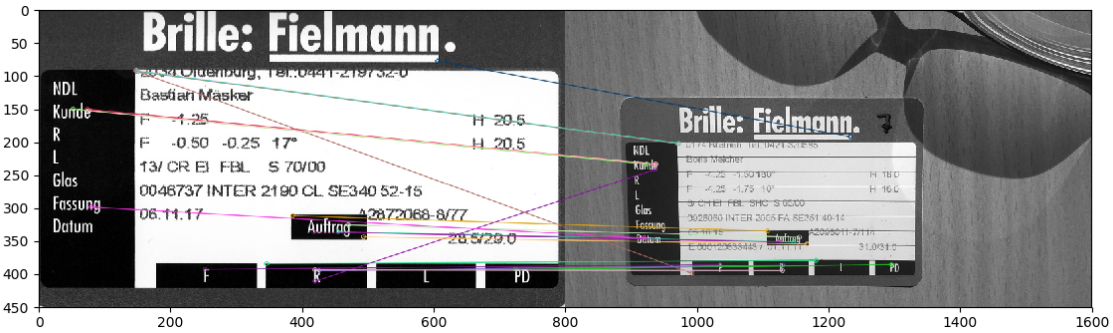


Abbildung 52: Visualisierung Feature Matching [Eigene Darstellung]

In Abbildung 52 ist das Feature Matching für die Punkte mit der geringsten Distanz

zwischen zwei verschiedenen Fielmann Pässen visualisiert. Diese Grafik veranschaulicht die Funktionsweise des Feature Matching und stellt die 20 kürzesten Distanzen dar.

BPL.2.4 Erkennung der gedruckten Werte Für das Auslesen der Sehwerte von den entsprechenden Brillenpässen wird im Allgemeinen folgendes Vorgehen verwendet. Im ersten Schritt müssen die Textelemente auf den Brillenpässen gefunden werden. Textelemente bezeichnen in diesem Kontext Wörter oder Zahlen, die die Informationen über den Kunden enthalten, welche auf den Brillenpässen notiert sind. In diesem Anwendungsfall wird die Einschränkung getroffen, dass nur die Sehwerte des Kunden relevant sind und weitere Informationen, wie Name oder Ausstellungsdatum, vernachlässigt werden können. Demnach besteht die Zielsetzung darin, die Textelemente der folgenden Linsenwerte, sofern vorhanden, ausfindig zu machen:

- Sphäre
- Zylinder
- Achse
- Pupillendistanz
- Addition

Nachdem die relevanten Textelemente ausfindig gemacht sind, werden diese im zweiten Schritt gecroppt und mittels einer Texterkennungssoftware ausgewertet. Sobald der Inhalt des Bildausschnittes extrahiert wurde, wird der entsprechende Text mit den zugehörigen Koordinaten abgespeichert und steht dann für die weitere Auswertung bereit. Um die Selektion der Textelemente auf den Brillenpässen zu ermöglichen, wurden im Verlauf des Projektes zwei Lösungsansätze entwickelt, welche im Folgenden genauer vorgestellt werden.

Texterkennung mittels MSER und OpenCV: Die erste Methode orientiert sich am Vorgehen von [VV15]. In dieser Publikation wird ein Lösungsansatz für die Texterkennung vorgestellt, der ohne maschinelles Lernen auskommt. In der Anfangsphase des Projektes wird zunächst diese Methodik implementiert, um der Applikation möglichst schnell Funktionali-tät zu geben.

Die Grundlage für diese Herangehensweise ist das Maximally Stable Extremal Regions Verfahren (MSER). Dieser Algorithmus wird im Bereich der Computer Vision eingesetzt, um charakteristische Bildregionen ausfindig zu machen und um diese beispielsweise mit anderen Bildausschnitten zu vergleichen. Dafür identifiziert dieses Verfahren relevante Features in Bildern und eben diese Funktion wird für dieses Projekt übernommen, um Buchstaben,

Zahlen und Zeichen auf Brillenpässen zu erkennen.

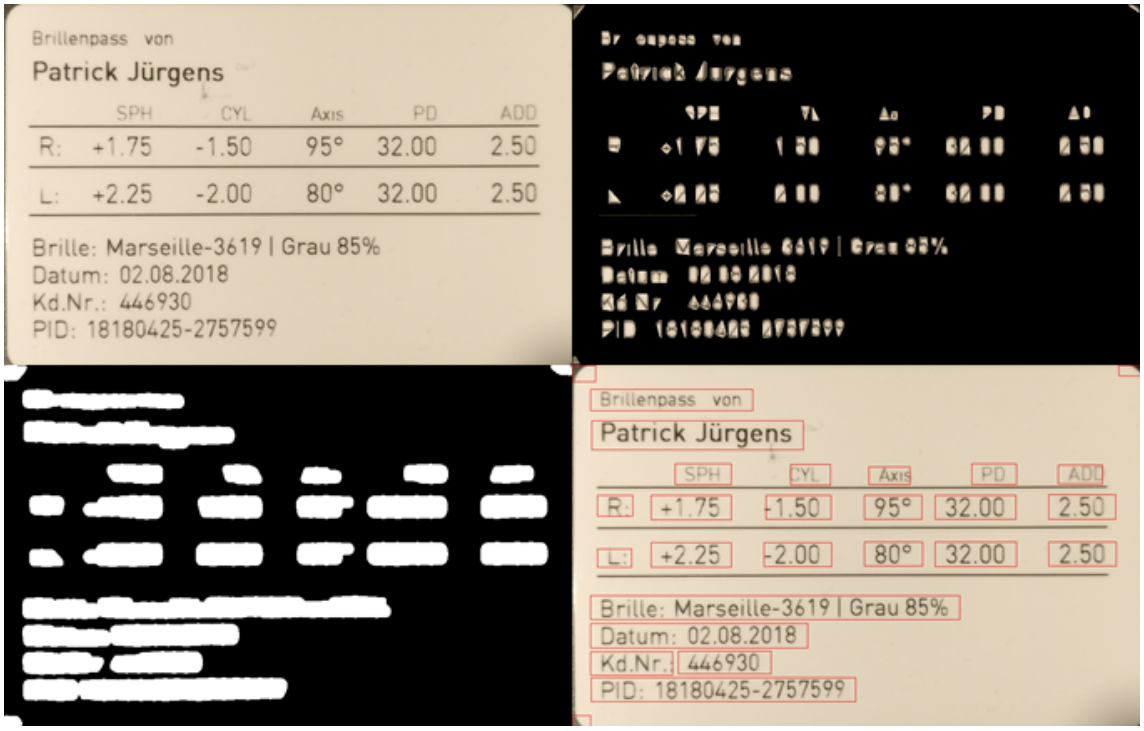


Abbildung 53: Arbeitsschritte beim MSER-Ansatz [Eigene Darstellung]

In Abbildung 53 ist das Resultat des MSER-Verfahrens im oberen rechten Bild dargestellt. Das Beispiel verdeutlicht, dass der Ansatz in der Lage ist, Buchstaben und Ziffern korrekt zu erkennen. Allerdings lassen sich dadurch nur einzelne Zeichen und keine zusammenhän-genden Wörter oder Zahlen erkennen. Dementsprechend muss dem MSER noch eine weitere Funktion nachgelagert werden. Dafür wird der Output aus dem MSER nicht mehr direkt ins Bild eingezeichnet, sondern in eine Maske. Das Ergebnis dieses Schrittes ist eine binäre Maske, in der die Konturen der Buchstaben und Ziffern eingezeichnet werden. Die entstandene Maske wird anschließend in horizontaler Richtung verzerrt, sodass eine neue Maske entsteht, in der benachbarte Zeichen zusammengefasst werden (vgl. Abbildung 53: unten links). Abschließend werden die Konturen der Objekte ermittelt und die Textelementen können ausgeschnitten werden. Die Koordinaten der entsprechenden Konturen werden gespeichert und später für die Zuordnung der Sehwerte verwendet.

Nachdem die Textelemente ausgeschnitten und mit den zugehörigen Koordinaten abgespeichert wurden, erfolgt das Auslesen des Textes. Für diese Aufgabe wird die Open

Source Texterkennungs-software Tesseract verwendet. Die Wahl fiel auf Tesseract, da, neben der freien Verfügbarkeit der Software, zusätzliche Einstellungen vorgenommen werden können. Bei Tesseract können beispielsweise die Vorhersagen so voreingestellt werden, dass nur einzelne Wörter erkannt werden. Durch diese Einstellungen kann die Qualität der Texterkennung deutlich erhöht werden. Im Anschluss an diesen Schritt ist der Text, der einzelnen Textelemente sowie deren zugehörige Koordinaten in einem Numpy-Array abgespeichert und steht dann für die Auswertung bereit (vgl. BPL.2.5).

Der erste hier vorgestellte Ansatz hat die Anforderungen nahezu vollständig erfüllt. Die Textelemente konnten selektiert und ausgewertet werden. Allerdings müssen dabei einige Einschränkungen beachtet werden. Zum einen ist die Qualität der Ergebnisse stark von der Qualität des aufgenommenen Fotos abhängig. Beispielsweise führen Spiegelungen des Sonnenlichts in reflektierenden Brillenpässen dazu, dass einzelne oder alle Sehwerte nicht zurückgegeben werden können. Darüber hinaus benötigt das MSER einen fehlerfreien Brillenpass. Das wird insbesondere bei Brillenpässen deutlich, die leichte Knicke aufweisen. Diese Unebenheiten werden vom MSER als Features erkannt und der Output ist dem-entsprechend unbrauchbar. Eine weitere Einschränkung beim MSER besteht darin, dass dieses Verfahren nicht filtern kann, ob es sich bei dem gefundenen Objekt um Text handelt. Als Folge dessen werden auch Bildausschnitte an Tesseract weitergegeben, die offensichtlich keinen Text enthalten und die Antwortzeit des Services wird dadurch negativ beeinträchtigt. Aufgrund dieser Umstände wird ein weiterer Ansatz entwickelt, der die hier beschriebenen Herausforderungen löst.

Texterkennung mittels CTPN: Aufgrund der aufgeführten Problematiken mit dem MSER-Ansatz, wird mit dem Connectionist Text Proposal Network (CTPN) eine zweite Herangehensweise entworfen. Diese orientiert sich an dem Vorgehen von [Tia+16] und verwendet maschinelles Lernen zur Identifikation der Textelemente.

Der Grundgedanke des vorgestellten Ansatzes wird durch die Architektur deutlich. Die Basis des CTPN bilden einige Convolutional Layer, welche der VGG-Architektur nachempfunden sind. Dementsprechend besteht auch das Grundgerüst des CTPN aus Blöcken, welche sich jeweils aus mehreren Convolutional Layern und einem anschließenden MaxPooling zusammensetzen. Im Gegensatz zu den bisherigen Anwendungen des Transfer-Learning wird dieses Modell nicht vor der letzten Schicht abgeschnitten, sondern bereits nach dem vierten Block. Durch dieses Verfahren wird das Eingangsbild in Teilbereiche aufgeteilt, die Rasterzellen genannt werden. In diesem Fall wird die Skalierung der ausgeschnittenen und

transformierten Brillenpässe auf eine einheitliche Größe gebracht (608, 1024, 3). Durch das mehrfache MaxPolling ist die finale Form der Matrix deutlich kleiner. In unserem Anwendungsfall verbleiben 38 Felder in der Höhe und 128 Felder in der Breite. Für jede dieser Rasterzellen wird anschließend eine Vorhersage getroffen, ob und wo sich Text in der entsprechenden Zelle befinden kann. Eine weitere Besonderheit des CTPN liegt darin, dass an dieser Stelle noch nicht der entsprechende Output erzeugt wird, wie es beispielsweise bei dem Yolo-Ansatz des CornerDetectors der Fall ist. Zwischen der letzten Schicht des VGG-Abschnittes und des letzten Convolutional Layer wird noch eine neue Netzebene eingefügt. Das verbindende Element ist ein Long Short-Term Memory (LSTM) Layer. Diese rekurrente Netzarchitektur wird verwendet, um den Sequenzcharakter des Zielobjektes besser abbilden zu können. Der Output des finalen Convolutional Layers besteht aus 38*128 Rasterzellen, die jeweils vier Parameter enthalten. Diese Parameter setzen sich aus den folgenden Werten zusammen:

- Mittelwert
- Höhe
- Side Refinement
- Prediction Score

Der Mittelwert und die Höhe geben an, wie das Textfeld innerhalb der Rasterzelle entlang der y-Achse positioniert werden soll. Analog dazu existiert noch ein Parameter für das sogenannte Side Refinement, wodurch das Textfeld entlang der x-Achse eingegrenzt werden kann. Dieser Wert liegt zwischen -1 und 1. Bei Werten, die ungleich 0 sind, wird der entsprechende Faktor von der Länge der Rasterzelle subtrahiert. Bei negativen Werten startet der Vorgang vom linken Rand der Rasterzelle und bei positiven Werten umgekehrt vom rechten Rand.

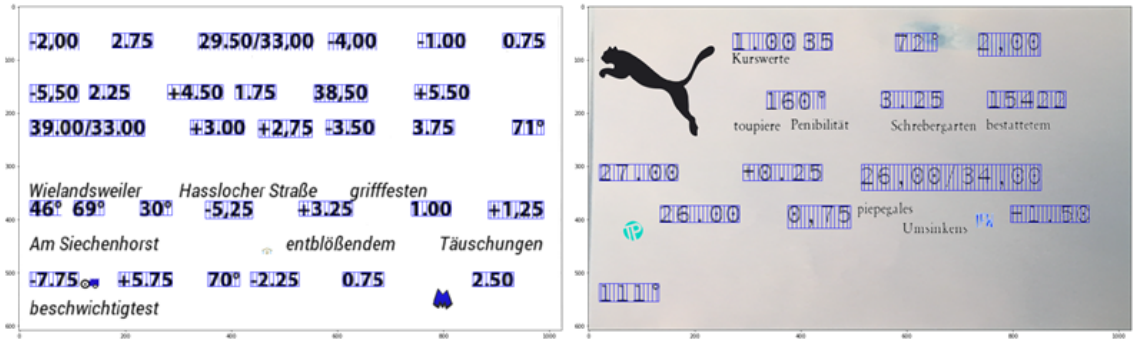


Abbildung 54: Trainingsdaten CTPN - Beispiele [Eigene Darstellung]

Im Gegensatz zum MSER-Ansatz besteht bei diesem Lösungsweg die Problematik, dass

Trainingsdaten für den Aufbau eines Modells benötigt werden. Da keine Datasets existieren, die für unseren Anwendungsfall ausgelegt sind, muss eine eigene Datengrundlage geschaffen werden. Während in vergleichbaren Situationen in anderen Teilprojekten, eigene Datasets durch annotieren aufgebaut werden, wird an dieser Stelle ein anderer Weg gewählt und ein Daten Generator entwickelt, welcher eine unendliche Menge an Trainingsdaten künstlich erzeugt. Der Daten Generator wählt aus einer Menge an Hintergründen zufällig einen aus und versieht diesen dann mit Logos und Text. Für die Logos wird jeweils ein Dataset für größere (190 Bilder) und eins für kleinere Logos erstellt (4979 Bilder). Die Logos enthalten keine Ziffern und es werden dann zufällig einige auf den Hintergründen platziert, um die Varianz der Trainingsdaten zu erhöhen. Die Logos erzeugen gerade zu Beginn des Trainings viele False-Positive-Resultate und das Netz kann durch ihre Anwesenheit stabilere Ergebnisse erzielen. Neben den Logos muss natürlich auch noch der Text auf den Bildern platziert werden. Zum einen sind das die entsprechenden Sehwerte. Dafür wird ein Skript geschrieben, welches für jeden der fünf Werte, zufällige Elemente zurückgibt, die realistische Werte wiederspiegeln. Zusätzlich wird die Schreibweise der Werte zufällig festgelegt. Beispielsweise kann eine Gleitkommazahl, je nach Herausgeber, durch einen Punkt oder ein Komma gekennzeichnet werden. In Listing 15 ist ein solches Skript für das Erstellen eines zufälligen Sphärenwertes als Beispiel aufgeführt. Diese Textblöcke werden mit den entsprechenden Annotationen versehen und stehen dann für das Preprocessing bereit. Neben den Sehwerten werden auch noch weitere Textsequenzen auf das Dokument geschrieben. Diese kennzeichnen Elemente, die das Netz später ignorieren soll und bestehen aus Wörtern oder irrelevanten Zahlenfolgen. Von diesen Elementen werden demnach auch keine Annotationen gespeichert. Zwei Beispiele für zufällig generierte Trainingsdaten finden sich in Abbildung 54. Für das Training des Modells steht demnach eine nahezu unendliche Menge an Trainingsdaten mit der entsprechenden Annotation zur Verfügung. Im Rahmen des Preprocessings müssen diese Angaben entsprechend umgerechnet werden, damit der Output zum Modell passt. Die Form der Ground Truth Werte für das Training ist 38*128*4 und dementsprechend ist für jede Rasterzelle eine Vorgabe mit 4 Parametern gegeben.

```
def create_sph():
  ran = round(((1 - random.random() * 2) * 10), 2)
  value = ran - ran % 0.25
  if value > 0:
    sph = "+" + '{0:.2f}'.format(value)
  else:
    sph = '{0:.2f}'.format(value)
  if random.random() > 0.5:
    sph = change_floating_point(sph)
return sph
```

Listing 15: Skript zur Erzeugung von Sphären-Werten

Für den Trainingsprozess wird ein Generator entworfen, der den Trainingsdatengenerator aufruft und die Daten für das Training bereitstellt. In diesem Generator werden die Daten augmentiert und es kann das Preprocessing durchgeführt werden. Sobald die Daten in der korrekten Form bereitstehen, können jeweils zwei Bilder mit Ground Truth Werten in einer Batch eingelesen werden. Aufgrund der Größe der Architektur ist eine größere Batchsize bei der gegebenen Rechenleistung nicht umsetzbar.

Bevor das Training starten kann, muss eine eigene Loss-Funktion entwickelt werden. Der erwartete Output kann nicht durch Loss-Funktionen bewertet werden, die Keras oder Tensorflow bereitstellen. Es muss eine spezielle Funktion angegeben werden, welche sich aus mehreren Bestandteilen zusammensetzt. Der CTPN-Loss ist in Listing 16 abgebildet und setzt sich aus drei Bestandteilen zusammen:

- **Loss1**: Der erste Teil gibt den Fehler des Prediction Scores an und wird durch die *Binary Crossentropy* angegeben.
- **Loss2**: In diesem Teil wird die Genauigkeit von Höhe und Mittelwert anhand des *Mean Squared Errors* bemessen.
- **Loss3**: Der letzte Teil der Loss-Funktion bewertet die Präzision des Side Refinements, ebenfalls anhand des *Mean Squared Errors*. Loss2 und Loss3 werden getrennt voneinander betrachtet, da sie unterschiedlich gewichtet werden (siehe *lamda_2* und *lambda_3* in Listing 16)

```
def ctpn_loss(y_true, y_pred):
  lambda_1, lambda_2, lambda_3 = 1.0, 1.0, 2.0

  obj_true = y_true[..., 2::4]
  coord_mean_true = y_true[..., 0::4]
  coord_height_true = y_true[..., 1::4]
  ref_true = y_true[..., 3::4]

  obj_pred = K.sigmoid(y_pred[..., 2::4])
  coord_mean_pred = y_pred[..., 0::4] * obj_true
  coord_height_pred = y_pred[..., 1::4] * obj_true
  ref_pred = y_pred[..., 3::4] * obj_true

  loss1 = lambda_1 * K.mean(K.binary_crossentropy(obj_true, obj_pred),
    axis=-1) / 3

  loss2 = lambda_2 * K.mean(K.abs(coord_mean_pred - coord_mean_true) + K.
    abs(coord_height_pred - coord_height_true), axis=-1) / 3

  loss3 = lambda_3 * K.mean(K.abs(ref_pred - ref_true), axis=-1) / 3

  loss = K.mean(loss1 + loss2 + loss3, axis=-1)

return loss
```

Listing 16: CTPN Loss-Funktion

Das Training beginnt mit einer Learning Rate von 0,0001, welche aus dem Grundlagenpaper hervorgeht. Für die Optimierung des Modells wird ein Callback eingerichtet, welcher die aktuelle Learning Rate nach zwei Epochen, in denen der Loss nicht sinken konnte, mit dem Faktor 0,9 multipliziert und dadurch reduziert. Da die Daten zur Laufzeit künstlich erzeugt werden, kann keine Epoche, im eigentlichen Sinne, definiert werden. Für die Verwendung von Callbacks und zum Überwachen des Trainings sind feste Epochen allerdings notwendig. Es werden 400 Bilder als eine Epoche festgelegt, um den Loss der Epochen vergleichbar zu machen. Nach 900 Epochen stellt sich ein stabiler Zustand ein und der Loss stagniert.

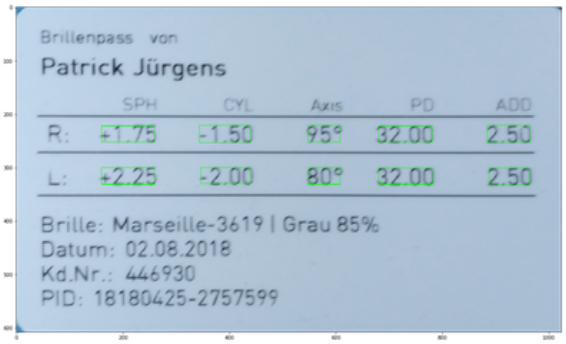


Abbildung 55: CTPN Ergebnis - Beispiel [Eigene Darstellung]

Der zweite vorgestellte Ansatz erfüllt die Anforderungen deutlich besser als die erste entwickelte Implementierung, welche das MSER-Verfahren verwendet. Die Ergebnisse sind deutlich robuster und demnach weniger anfällig für Probleme bei schlechten Aufnahmen oder einer schlechten Qualität des Brillenpasses. Dies wird insbesondere dann deutlich, wenn die Ergebnisse des MSER-Ansatzes aus Abbildung 53 mit den Ergebnissen des CTPN aus Abbildung 55 verglichen werden. Während im ersten Ansatz noch Elemente erkannt werden, die keinen Text enthalten (Ecken oder Unebenheiten), gibt der CTPN nur Elemente mit Text zurück. Zusätzlich werden nur die Textelemente berücksichtigt, welche mögliche Linsenwerte darstellen. Auch bei der Präzision der Koordinaten ist ein deutlicher Qualitätsgewinn erkennbar und die Textfelder können dadurch besser ausgeschnitten werden. Dies ist besonders dann nützlich, wenn der horizontale Abstand zwischen den Linsenwerten gering ist, wie es bei den Fielmann Brillenpässen der Fall ist.

BPL.2.5 Zuordnen der ausgelesenen Werte zu Eigenschaften Um die ausgelesenen Werte zuordnen zu können wird die Position des Fundes mit der Soll-Position aller Werte auf dem Brillenpass abgeglichen. Hierzu sind die Soll-Positionen aller Werte für Brille24 und Fielmann Brillenpässe in einem Array hinterlegt:

```
COORDS_BRILLE24 = np.array([["rSph", 178, 226],
                ["rCyl", 332, 226],
                ["rAxis", 492, 226],
                ["rPd", 624, 225],
                ["rAdd", 795, 225],
                ["lSph", 178, 298],
                ["lCyl", 332, 298],
                ["lAxis", 482, 298],
                ["lPd", 624, 299],
                ["lAdd", 794, 299]])

COORDS_FIELMANN = np.array([["rSph", 250, 230],
                ["rCyl", 350, 230],
                ["rAxis", 430, 230],
                ["rAdd", 570, 230],
                ["lSph", 250, 275],
                ["lCyl", 350, 275],
                ["lAxis", 430, 275],
                ["Pd", 820, 460],
                ["lAdd", 570, 275]])
```

Listing 17: Globale Positionswerte der Brillenpässe

Tesseract liefert beim auslesen der Werte sowohl den erkannten Text, als auch die Position des Fundes zurück. Anschließend wird mit der Methode *findlensvalues()* das Matching vorgenommen.

```
def findlensvalues(textblocks, preset: str):
  """
  Find lensvalues
  """
  if preset is Optician.FIELMANN.value:
    coordinates = COORDS_FIELMANN
  elif preset is Optician.BRILLE24.value:
    coordinates = COORDS_BRILLE24

  # Find lensvalues using the shortest Euclidean Distance
  lensvalues = np.array([["Description", "Value"]])
  maxeuclideandistance = 30
  for row_aim in coordinates:
    for row_found in textblocks:
      dist = sqrt((row_found[0].astype(np.float) - row_aim[1].astype(np.
float)) ** 2 + (
      row_found[1].astype(np.float) - row_aim[2].astype(np.float)) ** 2)
      if dist < maxeuclideandistance:
        lensvalues = np.append(lensvalues, [[row_aim[0], row_found[2]]],
axis=0)
```

Listing 18: Matching Soll/Ist-Positionen

Hierzu wird zu jedem gefundene Werte die euklidische Distanz zu den Soll-Positionen der Werte errechnet. Der gefundene Werte wird dem Sollwert mit der kürzesten Distanz (max. 30 Pixel) zugeordnet.

BPL.2.6 Rückgabe der ausgelesenen Werte in ein standardisiertes Datenformat Um die ausgelesenen Daten verarbeiten zu können ist es notwendig ein standardisiertes Datenformat einzuführen. Hierbei wird wie in der gesamten Kommunikation zwischen verschiedenen Diensten das JSON-Format eingesetzt. Das Schema für den Brillenpass muss alle möglichen Parameter eines Brillenpasses enthalten: Sphäre, Zylinder, Achse, Addition und Pupillendistanz.

BPL.2.7 Erzeugung von Testdaten Um festzustellen, welche Brillenpässe verarbeitet werden müssen, wurde eine kleine Marktanalyse durchgeführt. Diese hatte zum Ergebnis, dass Fielmann im Jahr 2017 der Marktführer war, mit einem Nettoumsatz von 1.085 Millionen Euro, gefolgt von Apollo Optik mit 694 Millionen Euro. Auf dem dritten Platz befand sich Pro Optik mit einem Nettoumsatz von 125 Millionen Euro. Dem folgen viele kleinere Optikerketten. Unser Augenmerk lag daher auf den Brillenpässen von Fielmann, Apollo und Brille24. Unsere Recherche ergab allerdings, das Apollo Optik keine Brillenpässe an

ihre Kunden ausstellt. Daher haben wir uns auf die Brillenpässe von Fielmann und von Brille24 beschränkt [Sta18].

Zum Testen des Brillenpasslesers wurde ein Brillenpassgenerator entwickelt. Dieser kann automatisch Brillenpässe von Fielmann oder Brille24 mit Zufallszahlen generieren. Zur Vorbereitung wurden Blanko-Vorlagen der Brillenpässe von Fielmann und Brille24 erstellt, indem die Werte ausradiert wurden, so dass diese leer erscheinen. Bei den Brillenpässen von Fielmann unterscheiden wir zudem zwischen neuen und alten Pässen. Diese unterscheiden sich lediglich dadurch, dass die „alten“ Pässe Trennlinien zwischen den einzelnen Werten haben und die „neuen“ diese nicht mehr aufweisen.

Funktionsweise: Der Passgenerator ermittelt einen zufälligen Vor- und Nachnamen aus einer Liste von ausgewählten Namen. Die Sphäre setzt sich zusammen aus einem Vorzeichen (+ oder -), einem Sphärenwert zwischen 0 und 7 und einer Endung in 0,25er Schritten. Dabei wurde darauf geachtet, dass der Sphärenwert möglichst realistisch gehalten wird, indem bevorzugt niedrige Werte ausgegeben werden. Analog dazu wurde mit dem Wert für den Zylinder verfahren. Dieser setzt sich aus Vorzeichen, Wert zwischen 0 und 3 sowie einer Endung zusammen. Die Achse wird berechnet aus einem Zufallswert zwischen 0 und 180 Grad. Außerdem wird die Pupillendistanz generiert. Diese setzt sich zusammen aus einem Wert zwischen 24 und 38 sowie einer Endung in 0.5er Schritten.

Mithilfe von OpenCV werden diese Werte dann auf den jeweiligen Brillenpass geschrieben. Zur Generierung eines Brillenpasses werden Koordinaten und Schriftgröße übergeben, um die Werte an die richtige Stelle in den verschiedenen Vorlagen einzufügen.

Durch eine Schleife kann die gewünschte Anzahl an Brillenpässen bestimmt werden. Diese werden dann mit aufsteigenden Nummern benannt und gespeichert.

11.4 Abnahme

Folgend wird die Erfüllung der einzelnen Anforderungen bewertet. Erfüllte Anforderungen werden mit **Abgenommen**, nicht erfüllte Anforderungen mit **nicht Abgenommen** markiert. Anforderungen, die auf andere Art und Weise gelöst wurden erhalten die Markierung **nicht umgesetzt**. Die Abnahme der Benutzeroberfläche ist in Tabelle 27 aufgeführt. Die Abnahme des Servicelayers folgt in Tabelle 28.

Nr.	Beschreibung	Abgenommen
BPL.1.1	Aufnahmen eines Fotos mit der Gerätekamera	**Abgenommen**
BPL.1.2	Verifizieren des aufgenommen Fotos durch den Nutzer	**Abgenommen**
BPL.1.3	Senden des Fotos an den HTTP-Endpunkt des Servicelayers	**Abgenommen**
BPL.1.4	Anzeigen der ausgelesenen Werte	**Abgenommen**
BPL.1.5	Ändern der ausgelesenen Werte	**Abgenommen**
BPL.1.6	Validieren der ausgelesenen Werte	**Abgenommen**
BPL.1.7	Rückgabe der Werte an einen beliebigen HTTP-Endpunkt	**Abgenommen**

Tabelle 27: Abnahme der Anforderungen an die Benutzeroberfläche

Nr.	Beschreibung	Abgenommen
BPL.2.1	Empfangen von Fotos	**Abgenommen**
BPL.2.2	Finden und ausschneiden des Brillenpasses	**Abgenommen**
BPL.2.3	Klassifizierung des Herausgebers	**Abgenommen**
BPL.2.4	Erkennung der gedruckten Werte	**Abgenommen**
BPL.2.5	Zuordnen der ausgelesenen Werte zu Eigenschaften (Sphere, Addition usw.)	**Abgenommen**
BPL.2.6	Rückgabe der ausgelesenen Werte in einem standardisierten Datenformat	**Abgenommen**
BPL.2.7	Erzeugung von Testdaten	**Abgenommen**

Tabelle 28: Abnahme der Anforderungen an den Servicelayer

11.5 Fazit

Abschließend kann festgestellt werden, dass alle Anforderungen an den Brillenpassleser umgesetzt wurden. Der Dienst liefert für Brille24- und Fielmann-Brillenpässe gute Ergebnisse. Insbesondere für die Brillenpässe von Brille24 sind die Ergebnisse auch robust. Bei der Anwendung des Dienstes auf Fielmann-Brillenpässe können jedoch gelegentlich Probleme auftreten. Diese treten bei der Zuordnung der ausgelesenen Werten zu den Eigenschaften auf (siehe BPL.2.5), obwohl alle Textfelder zuvor korrekt erkannt wurden. Die Probleme lassen sich mit dem noch unzureichenden Zuordnungsverfahren begründen. Die Position der Elemente auf dem Brillenpass ist bei Fielmann uneinheitlich und weicht teilweise deutlich von dem Standard ab, den wir für die Bestimmung der globalen Positionswerte verwendet haben. Zusammengefasst lässt sich sagen, dass das angewandte Verfahren nicht die benötigte Robustheit liefert, um zum aktuellen Zeitpunkt an Kunden ausgeliefert werden. Aufgrund der kurzen Entwicklungszeit sollte der Brillenpassleser als

Proof-of-Concept angesehen werden.

11.6 Ausblick

Um mit der aktuellen Version ein marktreifes Produkt zu erreichen, sollten zwei Bereiche des Brillenpasslesers überarbeitet werden. Zum einen müssen weitere Herausgeber identifiziert und zum Dienst hinzugefügt werden. Die Anzahl der Herausgeber, die mit dem bisherigen Verfahren bearbeitet werden können, ist allerdings begrenzt. Das Feature Matching kann dann nur eingeschränkt als Klassifizierungswerkzeug dienen. In diesem Fall sollte diese Herangehensweise durch eine klassische Klassifizierung durch neuronale Netze ersetzt werden, wie sie innerhalb dieses Projektes bereits mehrfach zum Einsatz gekommen ist. Ein weiterer Ansatzpunkt für die Verbesserung dieses Dienstes liegt in dem Zuordnungsverfahren. Dieses eignet sich für genormte Brillenpässe und liefert dann auch robuste Ergebnisse. Sobald ein Herausgeber allerdings eine nennenswerte Varianz in seinen Pässen zulässt, funktioniert dieses Verfahren nicht mehr konstant genug.

12 Brillentinder

Das Brillentinder Teilprojekt verfolgt die Idee, es Kunden zu ermöglichen mittels Wischgesten auszudrücken, ob ihnen eine dargestellte Brille gefällt oder nicht. Basierend auf den Entscheidungen sollen dem Kunden weitere Brillen vorgeschlagen werden, die seinem Geschmack entsprechen. Da das Konzept an die populäre Onlinedating-App *Tinder* angelehnt ist entstand der Name *Brillentinder*.

12.1 Beschreibung

Brillentinder soll es ermöglichen, den subjektiven Geschmack eines Kunden abzubilden, um die möglichst optimale Brille zu finden. Das bisher implementierte Konzept des Brillenberaters versucht anhand optischer Merkmale eine objektiv möglichst passende Brillen zu finden, der Geschmack eines Kunden muss sich allerdings nicht zwangsweise mit der objektiven Betrachtung decken. Durch die Abbildung des subjektiven Geschmacks entsteht eine neue Sichtweise, welche die folgende Mehrwerte mit sich bringt:

- Brillenvorschläge basierend auf dem Kundengeschmack
- Verbessertes Einkaufserlebnis
- Datengewinnung über den subjektiven Geschmack der Kunden
- Daten über Merkmalsausprägungen
- Daten über Verbindungen von Brillenmodellen

Diese Mehrwerte ermöglichen es dem Kunden ein neues Einkaufserlebnis anzubieten und schaffen gleichzeitig einen Mehrwert für den Projektpartner Brille24, da die gewonnenen Daten den individuellen Geschmack eines Kunden abbilden können.

Verbunden mit den vorhandenen Komponenten, wie dem Brillenberater, sind zudem später verschiedene Kombinationen denkbar. Der Brillenberater kann vor- oder nachgelagert eingesetzt werden, um den subjektiven Geschmack mit der objektiven Betrachtung zu kombinieren. Eine Kombination mit einer virtuellen Anprobe wäre ebenfalls denkbar. Diese Szenarien werden im Ausblick genauer beschrieben.

12.2 Anforderungen

Folgend werden die Anforderungen definiert. Teilweise wurden diese durch den Projektpartner Brille24 vorgegeben, andere wurden in einem gemeinsamen Workshop erarbeitet oder entstanden während der Entwicklung. Die jeweilige Herkunft wird direkt in der Anforderungsdefinition beschrieben.

Nr.	Beschreibung
BRT.1.1	**Entscheidungen durch Wischgesten** Ein Kunde soll mittels Wischgestes entscheiden, ob eine Brille seinem Geschmack entspricht oder nicht. Wische nach Links entspricht einem *Gefällt mir nicht*, wischen nach Rechts einem *Gefällt mir*. **Herkunft:** Erarbeitet im Workshop
BRT.1.2	**Entscheidungen durch Schaltfläche** Ein Kunde soll ebenfalls per Schaltfläche entscheiden, ob eine Brille gefällt oder nicht. **Herkunft:** Erarbeitet im Workshop
BRT.1.3	**Kundenentscheidungen beeinflussen die Brillenvorschläge** Die Entscheidung (Gefällt mir/Gefällt mir nicht) eines Kunden beeinflusst, welche Brille dem Kunden als nächstes vorgeschlagen wird. **Herkunft:** Erarbeitet im Workshop
BRT.1.4	**Gefällt mir-Angabe beeinflusst die Entscheidung in Richtung der angezeigten Brille** Wählt ein Kunde *Gefällt mir* werden die zukünftigen Vorschläge in Richtung der vorgeschlagenen Brille angepasst **Herkunft:** Erarbeitet im Workshop
BRT.1.5	**Gefällt mir nicht-Angabe beeinflusst die Entscheidung entgegen der Richtung der angezeigten Brille** Wählt ein Kunde *Gefällt mir nicht* werden die zukünftigen Vorschläge in Gegenrichtung der vorgeschlagenen Brille angepasst **Herkunft:** Erarbeitet im Workshop
BRT.1.6	**Speicherung der Kundenangaben** Für jeden Brillenvorschlag wird die Kundenreaktion gespeichert, hierbei ist die Zuordnung zwischen den Aktionen eines Kunden möglich **Herkunft:** Erarbeitet im Workshop

Tabelle 29: Anforderungen an das Brillentinder Teilprojekt

12.3 Umsetzung

Die Umsetzung erfolgt in zwei Komponenten. Die erste Komponente stellt das Frontend dar, welches analog zu allen Frontend-Komponenten in Angular entwickelt wurde. Die Darstellung der Brillenvorschläge erfolgt auf rechteckigen Karten, welche vom Nutzer per Drag-and-Drop nach Links oder Rechts gewischt werden können. Zusätzlich kann die Auswahl per Button erfolgen (BRT.1.1 und BRT.1.2). Die Entscheidung des Kunden wird an die zweite Komponente, das Backend, gesendet und dort verarbeitet.

Die technische Umsetzung basiert auf mathematischen Repräsentationen aller von Bril-

le24 unter der Eigenmarke vertriebenen Brillen. Die mathematischen Repräsentationen werden als Embeddings bezeichnet. Ein Embedding bildet ein Objekt mittels eines mehrdimensionalen numerischen Vektors ab. Die Dimensionen innerhalb eines Vektors haben typischerweise keine inhärente Bedeutung. Die Bedeutung entsteht erst durch die Entfernung zwischen den Vektoren [Ten19].

Die Embeddings der Brillen wurden durch den Projektpartner bereitgestellt. An dieser Stelle wird davon ausgegangen, dass ähnliche Brillen einen geringen und unähnliche einen großen Abstand zueinander haben. Basierend auf dieser Annahme wird dem Kunden zu Beginn eine zufällig im Raum ausgewählte Brille vorgeschlagen. Die erste Brille dient lediglich als Startpunkt, um sicherzustellen, dass sich in der Nähe weitere Brillen befinden, die dem Kunden vorschlagen werden können. Der Kunde sieht diese Brille nicht. Ausgehend von der ersten Brille wird die Distanz zu allen Brillen im Raum berechnet und zufällig eine Brille aus den 1% ähnlichsten vorgeschlagen.

Wählt der Kunde die Option *Gefällt mir* wird der Ankerpunkt auf die vorgeschlagene Brille verschoben und erneut die Entfernung zu allen Brillen errechnet, um weitere Vorschläge zu generieren (BRT.1.3 und BRT.1.4). Wählt der Kunde *Gefällt mir nicht* wird der Ankerpunkt in die exakt entgegengesetzte Richtung und um die Entfernung zwischen Ankerpunkt und vorgeschlagener Brille bewegt. Es spielt hierbei keine Rolle, ob der Ankerpunkt sich auf einer Brille befindet oder im freien Raum (BRT.1.3 und BRT.1.5).

Um die erzeugten Daten später nutzen zu können müssen diese persistiert werden. Hierzu wird jeder Vorschlag mit der Entscheidung des Kunden und dem neuen Ankerpunkt in eine Datenbank (CouchDB) geschrieben. Um die Datensätze eines Kunden miteinander verbinden zu können wird zu Beginn einer Sitzung eine einzigartige Correlation-ID erzeugt und mit den Datensätzen gespeichert (BRT.1.6). Für den Prozess der Speicherung wird der Datenbankdienst der Plattform verwendet.

12.4 Abnahme

Folgend wird die Erfüllung der einzelnen Anforderungen bewertet. Erfüllte Anforderungen werden mit **Abgenommen**, nicht erfüllte Anforderungen mit **nicht Abgenommen** markiert. Anforderungen die auf andere Art und Weise gelöst wurden erhalten die Markierung **nicht umgesetzt**.

Nr.	Beschreibung	Abgenommen
BRT.1.1	Entscheidungen durch Wischgesten	**Abgenommen**
BRT.1.2	Entscheidungen durch Schaltfläche	**Abgenommen**
BRT.1.3	Kundenentscheidungen beeinflussen die Brillenvorschläge	**Abgenommen**
BRT.1.4	Gefällt mir-Angabe beeinflusst die Entscheidung in Richtung der angezeigten Brille	**Abgenommen**
BRT.1.5	Gefällt mir nicht-Angabe beeinflusst die Entscheidung in Richtung der angezeigten Brille	**Abgenommen**
BRT.1.6	Speicherung der Kundenangaben	**Abgenommen**

Tabelle 30: Abnahme der Anforderungen an Brillentinder

12.5 Fazit

Alle aufgenommenen Anforderungen wurden erfüllt und erfolgreich abgenommen. Das Teilprojekt bietet neben dem Brillenberater und dem Onlineshop eine dritte Möglichkeit für Kunden ihre Wunschbrille zu finden. Der Tinder-Ansatz stellt hierbei ein neues Einkaufserlebnis dar, welches sich von einem herkömmlichen Onlineshop unterscheidet. Außerdem werden Daten über den individuellen Geschmack eines Kunden gesammelt.

12.6 Ausblick

Ausblickend bietet Brillentinder diverse Einsatzmöglichkeiten. Der Service kann mit dem Brillenberater gekoppelt werden, um aus den vom Berater vorgeschlagenen Brillen auszuwählen. Alternativ könnte die Auswahl der Brillen für den Berater im Vorfeld mittels Brillentinder eingeschränkt werden. Der Berater könnte dann aus den Brillen die dem Geschmack des Kunden entsprechen die Brille wählen, die anhand der Gesichtsmerkmale am besten passt. Eine weitere Möglichkeit ist die Kopplung mit einer virtuellen Anprobe. Hierbei könnte der Kunde die vorgeschlagenen Brillen direkt mithilfe einer Gerätekamera virtuell anprobieren. Auch der Einsatz von Brillentinder bei der Registrierung eines Kunden wäre möglich, um direkt den individuellen Geschmack des Kunden abzubilden. Mit diesen Daten können dem Kunden dann direkt passende Brillen im Shop vorgeschlagen werden.

Auch einige technische Erweiterungen sind zukünftig möglich. Die Einführung eines Momentum in der Navigation könnte dabei helfen schneller die Wunschbrille zu finden. Bei dem Momentum-Ansatz würde die Länge der Schritte in der Navigation zwischen den Brillen zusätzlich durch die letzten Reaktionen beeinflusst. Wählt der Kunde mehrmals

hintereinander *Gefällt mir* oder *Gefällt mir nicht* verstärkt sich die Schrittlänge in die entsprechende Richtung. Eine weitere mögliche Anpassung wäre eine Beschränkung auf eine gewisse Anzahl an Durchgängen. Hierdurch würde der Kunde am Ende der Durchgänge eine Brille vorgeschlagen bekommen und kann nicht, wie aktuell, unendlich lange durch den Raum navigieren. Schwierig hierbei ist die Auswahl eines passenden Kriteriums. Eine Beschränkung nach Durchgängen hat den Nachteil, dass der Kunde möglicherweise noch kein befriedigendes Ergebnis erreicht hat. Alternativ wäre es denkbar das Momentum mit einzubeziehen und die Sitzung bei einem gewissen Momentum zu beenden.

13 Evaluation

In diesem Kapitel geht es um die Evaluation des Teilprojekts Brillenberater. Angefangen wird mit der Begründung für die Auswahl des zu evaluierenden Teilprojekts. Danach wird die Motivation für die Evaluation erläutert und die Anforderungen an das Evaluationsverfahren beschrieben. Da es mehrere Ideen für das Evaluations-Verfahren gab, werden diese vorgestellt und deren Eignung für die Evaluation des Brillenberaters geprüft. Nach der Festlegung auf ein Verfahren wird die Konzeption und abschließend die Durchführung und Auswertung beschrieben.

13.1 Auswahl des zu evaluierenden Teilprojekts

Da in der Projektgruppe mehrere Teilprojekte bearbeitet wurden, die jedoch nicht gemeinsam evaluiert werden können, war es nötig sich auf ein Teilprojekt für die in diesem Abschnitt beschriebene Evaluation festzulegen. Die Wahl fiel auf den Brillenberater, da eine (umfangreichere) Evaluation sich hier aus mehreren Gründen als am geeignetsten darstellte:

- **Bedeutung für die Projektgruppe**
  Da das Teilprojekt über die ganze Projektlaufzeit umgesetzt wurde und die wichtigste Anforderung seitens Brille24 angegangen wurde, nämlich die individuelle Brillenberatung unter Zuhilfenahme von KI, kommt dem Brillenberater die größte Bedeutung innerhalb der Projektgruppe zu.

- **Anzahl der (zukünftigen) Nutzer**
  Der Brillenberater ist darauf ausgelegt durch Endkunden genutzt zu werden und diesen einen Mehrwert zu bieten, sodass sich aufgrund der zu erwartenden Nutzungszahlen ein bedeutender Erkenntnisgewinn durch die Evaluation erwarten lässt. Im Gegensatz dazu sind die anderen Teilprojekte nur auf wenige User ausgelegt (Social Media Trendanalyse) oder sind nicht über den Status eines Proof of Concept hinweggekommen (Brillenpassleser und Brillentinder), sodass eine Evaluation nur einen begrenzten Nutzen gehabt hätte.

Eine Evaluation wurde dennoch für die anderen Teilprojekte durchgeführt, allerdings geschah dies nicht innerhalb einer dedizierten Evaluationsphase, sondern während der Umsetzung der Teilprojekte unter anderem in Form von Lasttests oder Interviews mit den Stakeholdern.

13.2 Motivation

Laut Trepper [Tre15, S.15] soll mit der Evaluation überprüft werden, ob sich die entwickelten Artefakte zur Lösung einer bestimmten Problemstellung eignen.
Die Problemstellung, welche der Entwicklung des Brillenberaters zugrunde lag war die, dass es für den Kunden mitunter schwer sein kann im Onlineshop die optimale Brille zu finden. Das Sortiment ist im Gegensatz zum stationären Handel sehr groß und die passende Brille zu finden ist oft mit einem großen Zeitaufwand verbunden. Bei der Wahl der Brille sollte zudem auf bestimmte Kriterien geachtet werden, wie z. B. eine Brillenform, die zum jeweiligen Gesicht passt. Auch das Geschlecht oder die Hautfarbe spielen bei der Brillenwahl eine Rolle. Der Brillenberater soll es dem Kunden vereinfachen eine passende Brille zu finden, in dem er zu ihm passende Brillen auf Basis eines Fotos seines Gesichts vorgeschlagen bekommt. Für weitere Informationen wird auf das Kapitel 9.1 verwiesen.
Für die Evaluation des Brillenberaters musste daher ein Verfahren gefunden werden mit dem geprüft werden konnte inwiefern der Brillenberater tatsächlich in der Lage ist, dem Kunden die Suche nach einer passenden Brille zu vereinfachen bzw. Brillen vorschlägt, die dem Nutzer gefallen.

13.3 Anforderungen an das Evaluationsverfahren

Die Intention und damit wichtigste Anforderung in Bezug auf die Evaluation ist, wie bereits in der Einführung erwähnt, die Prüfung auf die Eignung zur Problemlösung. Um ein valides Ergebnis erzielen zu können, waren hinsichtlich der Umsetzung weitere Faktoren zu beachten. So war eine möglichst zeitnahe Umsetzung nötig, da die Evaluation in der letzten Phase der Projektgruppe implementiert, durchgeführt und umgesetzt werden musste. Aufgrund des knappen zeitlichen Rahmens musste der Aufwand zudem möglichst gering gehalten werden. Auch sollten Abhängigkeiten von Dritten möglichst vermieden werden, um Verzögerungen zu verhindern. Zusammengefasst lauteten die Anforderungen für die Festlegung auf ein Evaluationsverfahren:

- Das Verfahren muss geeignet sein die Eignung zur Problemlösung zu prüfen
- Der Implementierungsaufwand sollte möglichst gering sein
- Abhängigkeiten von Dritten sollten klein gehalten werden

Die nachfolgende Festlegung auf ein Verfahren basierte auf diesen Anforderungen.

13.4 Festlegung auf ein Verfahren

Für die Umsetzung der Evaluation des Brillenberaters gab es mehrere Ideen, welche im Dezember in einem Meeting mit den Betreuern und einigen Teilnehmern der PG besprochen wurden. Es sollte ein Verfahren gefunden werden, welches den im vorigen Abschnitt definierten Anforderungen weitestgehend entsprach. Folgende Verfahren wurden diskutiert:

a) Freischalten des Brillenberaters für einen Teil der User im Onlineshop, um die Conversion Rate von altem und neuem Brillenberater gegenüberstellen zu können.

b) Live-Test des Brillenberaters mit anschließendem Interview

c) Nutzung des Brillenberaters und anschließende Bewertung des Ergebnisses durch die Probanden

d) Umfrage zur Zielgruppenidentifizierung

13.4.1 Bewertung des ersten Verfahrens

Bei diesem Verfahren sollte der neue Brillenberater nur für einen bestimmten Teil der Nutzer im Onlineshop von Brille24 freigeschaltet werden und es sollte ermittelt werden inwieweit sich die Conversion Rate durch den Brillenberater verändert.

- **Das Verfahren muss geeignet sein die Eignung zur Problemlösung zu prüfen**
  Das erste Evaluationsverfahren eignete sich in Bezug auf diese Anforderung sehr gut, da das Kaufverhalten der echten User direkt analysiert werden kann ohne sie zu beeinflussen. Es ist möglich zu ermitteln, ob User häufiger einen Kauf tätigen, wenn diese den Brillenberater verwendet haben. Da das ganze im Onlineshop stattfindet und es sich um reale Kunden handelt, würde dieses Verfahren sehr objektive Ergebnisse liefern.

- **Der Implementierungsaufwand sollte möglichst gering sein**
  Der Implementierungsaufwand ist vergleichsweise hoch, da Anpassungen im Live-Onlineshop notwendig wären. Des Weiteren wären sehr viele Absprachen mit dem Praxispartner notwendig. Der Aufwand ist somit als sehr hoch einzuschätzen, sodass diese Anforderung nicht erfüllt wird.

- **Abhängigkeiten von Dritten sollten klein gehalten werden**
  Da dieses Verfahren Anpassungen im Live-Onlineshop notwendig macht, wären Absprachen mit dem Praxispartner notwendig.

13.4.2 Bewertung des zweiten Verfahrens

Bei diesem Verfahren sollte der Proband den Brillenberater testen und im Anschluss daran in einem Interview seine Erfahrungen schildern und die Funktionen bewerten. Das Interview sollte in Form einer qualitativen Inhaltsanalyse durchgeführt werden.

- **Das Verfahren muss geeignet sein die Eignung zur Problemlösung zu prüfen**
  Das zweite Verfahren eignete sich in Bezug auf diese Anforderung nicht so gut wie das erste Verfahren, da der Proband hier subjektiv hätte beeinflusst werden können. Die Antworten im Interview könnten nicht dem Verhalten entsprechen, dass der Nutzer an den Tag legt, wenn er nicht unter Beobachtung steht. Des Weiteren könnten die Probanden durch die Fragestellung zu bestimmten Antworten gelenkt werden oder durch den Befragenden bei der Beantwortung beeinflusst werden. Die Anforderung wird mit diesem Verfahren nur bedingt erfüllt.

- **Der Implementierungsaufwand sollte möglichst gering sein**
  Für dieses Verfahren muss ein Fragebogen bzw. ein Interviewleitfaden konzipiert werden. Eine technische Umsetzung ist nicht notwendig. Der Aufwand ist eher gering, so dass die Anforderung erfüllt wird.

- **Abhängigkeiten von Dritten sollten klein gehalten werden**
  Es besteht keine Abhängigkeit von Dritten.

13.4.3 Bewertung des dritten Verfahrens

In diesem Verfahren sollte der Proband nach Abschicken seines Fotos mehrere Brillen bewerten von denen eine die ist, die der Brillenberater vorschlägt. Die anderen Brillen, die er angezeigt bekommt sind festgelegte oder zufällig ausgewählte Brillen. Ziel dieses Verfahrens ist es zu ermitteln, ob die durch den Brillenberater vorgeschlagene Brille besser bewertet wird, als zufällig ausgewählte oder vorher festgelegte Brillen.

- **Das Verfahren muss geeignet sein die Eignung zur Problemlösung zu prüfen**
  Das Verfahren ist darauf ausgerichtet die Brillenvorschläge des Brillenberaters durch den Probanden zu evaluieren. Im Vordergrund steht das subjektive Empfinden des Probanden zur vorgeschlagenen Brille im Vergleich mit zufällig ausgewählten Brillen. Das Verfahren eignet sich gut um die Vorschläge des Brillenberaters zu bewerten und zu prüfen, ob der Brillenberater für künftige Nutzer einen Mehrwert generiert.

- **Der Implementierungsaufwand sollte möglichst gering sein**
  Je nach gewünschter Umsetzungsform (mit Online-Anprobe oder ohne) ist mit einem mittleren Implementierungsaufwand zu rechnen.
- **Abhängigkeiten von Dritten sollten klein gehalten werden**
  Es bestehen keine Abhängigkeiten von Dritten.

13.4.4 Bewertung des vierten Verfahrens

Mit diesem Verfahren sollte mittels einer Umfrage via Online-Fragebogen eine Zielgruppenidentifizierung durchgeführt werden. Es sollten personenbezogene Daten erfasst werden, wie das Alter, Geschlecht und ob eine Sehschwäche besteht. Des Weiteren sollten Präferenzen in Bezug auf das (Online-)Kaufverhalten abgefragt werden sowohl allgemein als auch speziell in Bezug auf den Brillenhandel. Schlussendlich soll der Proband den Brillenberater anhand einer kurzen Beschreibung oder eines Videos vorgestellt bekommen und seine persönliche Einschätzung dazu geben. Der erste Entwurf für einen Fragebogen, welcher letztendlich nicht für die Evaluation verwendet wurde, ist im Anhang A.7 zu sehen.

- **Das Verfahren muss geeignet sein die Eignung zur Problemlösung zu prüfen**
  Der Fragebogen ist geeignet, um die Idee zu evaluieren und zu ermitteln, welche Zielgruppen von dem Tool besonders angesprochen werden. Das Artefakt, welches zur Problemlösung entwickelt wurde, wird jedoch nicht direkt evaluiert. Das Verfahren alleine liefert somit keine Ergebnisse, die das Endresultat bewerten und sollte nur in Kombination mit einem der anderen Verfahren eingesetzt werden.
- **Der Implementierungsaufwand sollte möglichst gering sein**
  Ein Online-Fragebogen lässt sich mittels diverser Online-Tools erstellen wie Google Formulare oder innerhalb von StudIP. Der Implementierungsaufwand ist somit sehr gering.
- **Abhängigkeiten von Dritten sollten klein gehalten werden**
  Es besteht keine Abhängigkeit von Dritten.

13.4.5 Gesamtbewertung und Festlegung auf ein Verfahren

Entschieden wurde sich für das dritte Verfahren, der Bewertung der Brillenvorschläge durch den Brillenberater, in Kombination mit dem vierten Verfahren, einem Online-Fragebogen. Die Entscheidung fiel auf diese beiden Verfahren, da die Bewertung der Bril-

lenvorschläge am geeignetsten schien, um die Eignung zur Problemlösung zu prüfen und durch den Fragebogen lassen sich zusätzliche für die Auswertung relevante Informationen erfassen um bspw. die Bewertungen der Brillen im Zusammenhang mit bestimmten Altersgruppen, dem Geschlecht oder dem Einkaufsverhalten zu setzen.

Da die anderen beiden Verfahren vom zeitlichen Aufwand in Bezug auf die Implementierung und der Auswertung nicht umsetzbar waren, wurde die Festlegung ohne Anwendung einer Methode, wie der Nutzwertanalyse oder Ähnliches, in einem Meeting durchgeführt.

13.5 Geplante Umsetzung

Wie im vorigen Kapitel beschrieben, sollte das Evaluationsverfahren eine Kombination aus Fragebogen und der Bewertung der Brillenvorschläge durch den Brillenberater werden. Für die Implementierung der Evaluation musste im Vorfeld definiert werden, wie der Ablauf gestaltet werden soll. Die Idee war eine Online-Umfrage durchzuführen mit anschließendem Test des Brillenberaters. So sollte der Proband erst ein paar Fragen zur Person beantworten und im Anschluss daran ein Foto mit dem Brillenberater machen. Der Proband sollte daraufhin nacheinander Brillen angezeigt bekommen, die er auf einer Skala von 0 bis 5 bewerten muss, wobei 0 den schlechtesten und 5 den besten Wert darstellt. Von den Brillen, die der Proband bewertet, ist eine die, die der Brillenberater empfiehlt und die anderen werden zufällig ausgewählt. Nachdem der Proband alle Brillen bewertet hat, werden ihm noch Fragen zum Brillenberater gestellt. Ziel der Evaluation sollte es sein festzustellen, ob die durch den Brillenberater empfohlene Brille im Durchschnitt besser bewertet wird, als eine zufällig ausgewählte. Der geplante Ablauf wird in Abbildung 56 veranschaulicht.

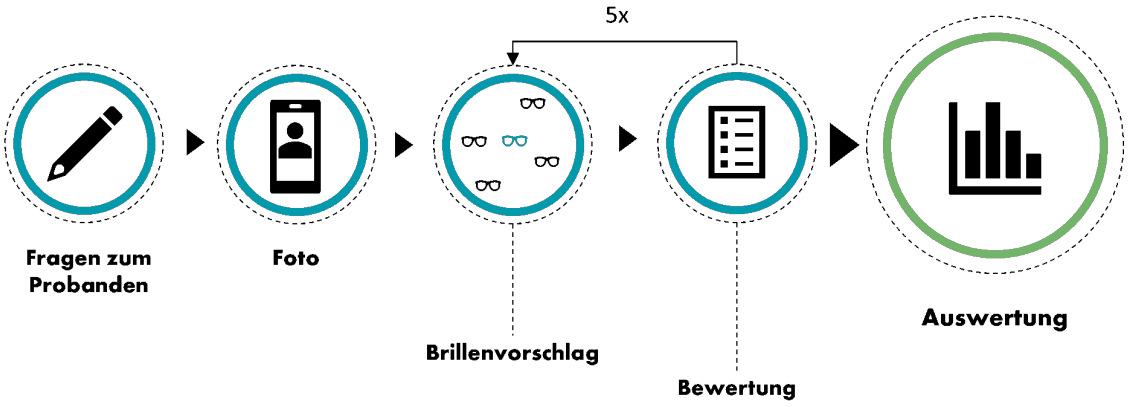


Abbildung 56: Ablauf der Evaluation (Quelle: Eigene Darstellung)

Die Fragen, die der Proband vor der Bewertung der Brillen beantworten musste, waren folgende:

- Alter
  Die Ergebnisse lassen sich durch diese Angabe in Bezug zu Altersgruppen setzen. Des Weiteren sollte diese Angabe genutzt werden, um die Erkennung des Geschlechts durch den Brillenberater zu evaluieren.

- Geschlecht
  Diese Angabe sollte zudem ebenfalls genutzt werden um die Vorhersage des Geschlechts durch den Brillenberater zu evaluieren.

- Sind Sie Brillenträger?
  Brillenträger sind die wichtigste Zielgruppe in Bezug auf den Brillenberater, sodass bei der Auswertung ein besonderes Augenmerk auf diese Nutzergruppe gelegt werden sollte.

- Haben sie schon eine Brille online gekauft?
  Es sollte festgestellt werden, ob der Proband bereits Erfahrungen gemacht hat in Bezug auf den Online-Brillenkauf, um diese Information ggf. in der Auswertung nutzen zu können.

- Könnten Sie sich vorstellen eine Brille online zu kaufen?
  Durch diese Frage sollte die generelle Akzeptanz des Probanden für den Online-Brillenkauf festgestellt werden.

Nach der Bewertung der Brillenvorschläge wurden dem Probanden Fragen zur Nutzung des Brillenberaters gestellt. Diese bezogen sich darauf, ob der Proband den Brillenberater in Zukunft nutzen würde und ob er sich beim Brillenkauf eher für den stationären Optiker oder für den Online-Brillenberater entscheiden würde.

- Könnten Sie sich vorstellen den Brillenberater in Zukunft zu nutzen?
  Der Proband konnte einen Wert auf einer Skala von 0-5 angeben, wobei 0 bedeutete, dass er sich gar nicht vorstellen kann den Brillenberater zu nutzen.

- Würden Sie eher den stationären Optiker oder den Online-Brillenberater nutzen?
  Hier handelte es sich um eine binäre Frage, bei der eine der beiden Optionen ausgewählt werden musste.

13.6 Technische Umsetzung

Für die technische Umsetzung der Evaluation wurde das Angular-Projekt des Brillenberaters kopiert und als eigenes Projekt mit eigener Registerkarte dem Frontend hinzugefügt. Dabei wurden einige Komponenten des Brillenberaters entfernt, modifiziert oder um neue Komponenten erweitert.

Entfernt wurde zum einen in der guide-home-component.ts der Button zum Hochladen eines Bildes, da der Nutzer der Evaluation lediglich die Möglichkeit haben soll, ein Bild aufzunehmen, sowie der beschreibende Begrüßungstext der Komponente, da sie nicht mehr als Startseite verwendet wird. Außerdem wurden einzelne Komponenten nicht direkt entfernt, aber werden nicht mehr angesprochen, da sie durch neu entstandene Evaluationskomponenten ersetzt wurden. Dies betrifft die Komponenten guide-result und guide-fileupload.

Modifiziert wurde hingegen zum einen die guide-home-Komponente. Neben den bereits erwähnten Entfernungen wurde ebenfalls der Textinhalt und das CSS für die Seite so angepasst, dass es die Evaluation unterstützt. Zum Anderen wurde die nextPage-Funktion der übergeordneten guide.component.ts angepasst, welche für die Weiterleitung zu anderen Komponenten dient. Die Funktion wurde so umgestellt, das ein Durchlauf durch die neu erstellten Evaluationskomponenten und noch bestehenden Brillenberaterkomponenten gewährleistet wird.

Neu hinzugekommen sind mehrere Komponenten, die die Durchführung der Evaluation vervollständigen. Dies gilt für folgende Komponenten:

- GuideEvaluationStartComponent: Sie dient für die Begrüßung des Nutzers und um erste Fragen zur Person selbst zu stellen, damit ein Überblick über Alter, Geschlecht und erste Kontaktpunkte mit online Brillenhandel gewonnen werden kann. Alter und Geschlecht helfen zudem dabei, die später ermittelten Werte der neuronalen Netze auf Korrektheit zu prüfen.
- GuideEvaluationResultsComponent: Hier werden nun zufällig ausgewählte Brillen den Nutzer vorgeschlagen. Hierfür wird eine zufällige Zahl errechnet. Der Bereich zwischen dem die zufällige Zahl liegen kann, bezieht sich dabei auf die Anzahl der noch verbleibenden Brillen in der Evaluation. Wird eine 0 dabei gewürfelt, wird die vom Brillenberater vorgeschlagene Brille gewählt. Bei einem anderen Wert oder wenn bereits eine 0 gewürfelt wurde, wird eine zufällige Brille entweder von unten oder von oben aus einem Array mit festgelegten Brillen vorgeschlagen. Die Entscheidung,

ob von oben oder unten gewählt wird, hängt dabei von der zufällig gewürfelten Zahl ab. Ist diese größer als die Hälfte der Anzahl der noch verbleibenden Brillen, wird von oben auswählt, andernfalls von unten. Der Nutzer der Evaluation sieht dabei die vorgeschlagene Brille und kann mittels Radio Buttons im Bereich von 0 bis 5 wählen, wie gut ihm die Brille gefällt. Klickt er auf weiter, wird ihm die nächste, nach dem oben beschriebene Verfahren zufällig gewählte, Brille angezeigt. Die Auswahl der Radio Button wird dabei zurückgesetzt. Sollten alle Brillen einmal vorgeschlagen worden sein, wird der Nutzer auf die nächste Komponente weitergeleitet.

- GuideEvaluationEndingComponent: Diese Komponente erscheint nach der Bewertung der Brillen. Der Nutzer wird dabei gefragt, wie zufrieden er mit dem Brillenberater ist und ob er sich eher für den Brillenberater oder einen stationären Optiker entscheiden würde. In beiden Fällen kann er dies mit Radio Buttons im Bereich von 0 bis 5 bewerten.

- GuideEvaluationFinishComponent: Hierbei handelt es sich um die letzte Komponente. Es wird sich dabei lediglich von dem Nutzer der Evaluation verabschiedet und für die Teilnahme bedankt.

Die Daten aus der Evaluation werden dabei während der einzelnen Schritte in einem Objekt gespeichert. Ist die Evaluation durchgegangen und alle Information aufgenommen, wird das Objekt als JSON-Objekt an die Datenbank geschickt und für weitere Analysen bereitgestellt.

13.7 Durchführung

Die Durchführung der Evaluation war für den März geplant, da die Entwicklung des Brillenberaters bis dahin abgeschlossen sein sollte. Geplant war eine durch die Gruppenmitglieder betreute Evaluation während der Volksbankmesse am 23.03.2019 mit Anprobe der vorgeschlagenen Brillen. Um die Anprobe der vorgeschlagenen Brillen zu ermöglichen, war es nötig die entsprechenden Brillengestelle durch Brille 24 zu bestellen. Hierfür mussten im Vorfeld 35 Brillen herausgesucht werden, die jedem Brillentyp entsprechen (rund, eckig, oval, einfarbig, mehrfarbig u. s. w.). Für die Evaluation wurde geplant den Probanden fünf Brillen vorzuschlagen, wobei eine die empfohlene vom Brillenberater sein sollte. Die vier Vergleichsbrillen wurden im Vorfeld festgelegt und nicht, wie in der Konzeptionsphase geplant, zufällig ausgewählt. Hintergrund war, dass für die anschließende Auswertung eine feste Vergleichsbasis gegeben sein sollte. Zwei der vier vorgeschlagenen Vergleichsbrillen waren für alle Probanden dieselben und die anderen zwei waren geschlechtsspezifische

Abbildung 57: Vergleichsbrillen für männliche Probanden (Quelle: Brille24 Onlineshop)

Brillen, die je nach erkanntem Geschlecht angezeigt wurden. Insgesamt waren somit sechs der 35 Brillen Vergleichsbrillen, sodass dem Brillenberater letztendlich 29 Brillen für die individuelle Empfehlung zur Verfügung standen. Die vier Vergleichsbrillen, die männlichen Probanden angezeigt wurden, sind in Abbildung 57 zu sehen. Da der Stand der Projektgruppe auf der Volksbank-Messe einen größeren Zulauf hatte, als erwartet, konnte die betreute Evaluation nicht wie geplant durchgeführt werden, sodass lediglich wenige Teilnahmen verzeichnet werden konnten. Aus diesem Grund wurde entschieden den Link zur Evaluation zu verteilen, um eine möglichst hohe Teilnehmerzahl zu generieren. Die Verteilung des Links geschah über eine Ankündigung in StudIP und das Auslegen eines durch die Projektgruppe erstellten Flyers. Des Weiteren wurde der Link innerhalb der Freundes- und Bekanntenkreise der Projektgruppenmitglieder verteilt (Siehe Abbildung 58).

13.8 Auswertung und Fazit

Die Durchführung der Evaluation wurde am 3.4.2019 beendet und es wurde direkt mit der Auswertung der Daten begonnen. Die Daten lagen im JSON Format vor und mussten für die Auswertung entsprechend aufbereitet werden. Es gab ein paar unvollständige Datensätze, die entfernt wurden und auch die Zuordnung der Bewertungen für die vom Berater vorgeschlagenen Brillen musste vereinheitlicht werden. Da die Brillen bei der Umfrage in zufälliger Reihenfolge angezeigt worden sind, befanden sich die Brillenvorschläge des Beraters in verschiedenen Spalten bzw. waren in der JSON nicht demselben Schlüssel zugeordnet. Nachdem die Aufbereitung der Daten abgeschlossen war, konnte mit der Auswertung begonnen werden. Hierfür wurde der Einfachheit halber Excel verwendet und die JSON Datei in Excel überführt. Insgesamt haben 103 Personen an der Evaluation teilgenommen, von denen vier die Umfrage nicht abgeschlossen haben, sodass diese Datensätze entfernt wurden. Es konnte somit auf 99 valide Datensätze zurückgegriffen werden. Wie in Abbildung 59 zu erkennen, waren mit 61,62% die meisten Probanden männlich und mit 64,65% waren auch fast zwei Drittel der Befragten Brillenträger. Lediglich 13,13% der Probanden haben bereits eine Brille gekauft, aber der Großteil der Befragten (86,87%) kann sich vorstellen eine Brille online zu kaufen. Das Geschlecht konnte der Brillenberater in

Abbildung 58: Frontansicht des Flyers für die Bewerbung der Evaluation (Quelle: Eigene Darstellung)

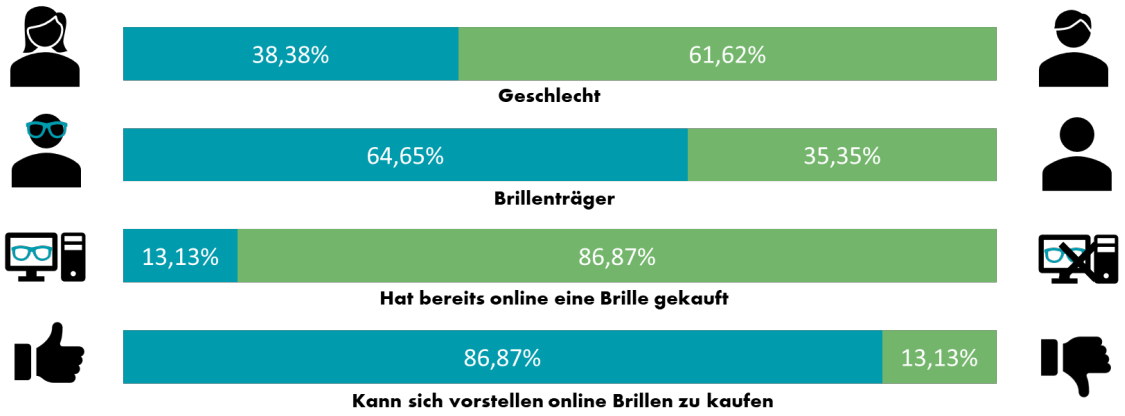


Abbildung 59: Ergebnisse der Anfangsbefragung (Quelle: Eigene Darstellung)

98,98% der Fälle richtig erkennen und das Alter mit einer durchschnittlichen Abweichung von 5,28 Jahren vorhersagen (Median: 4 Jahre).
Die vorgeschlagenen Brillen wurden bewertet nachdem der Proband die Anfangsfragen beantwortet und ein Foto gemacht hatte. Jede Brille musste auf einer Skala von 0-5 bewertet werden, wobei 5 den besten Wert darstellte. Die durch den Brillenberater vorgeschlagene Brille konnte eine Durchschnittsbewertung von 2,41 erzielen und lag damit 0,54 Punkte über der Durchschnittsbewertung der Vergleichsbrillen. Die Gegenüberstellung und prozentuale Verteilung der Bewertungen lässt sich in Abbildung 60 nachvollziehen.

In den abschließenden Fragen gaben 69,7% der Befragten an einen stationären Optiker dem Brillenberater vorzuziehen. Bei der Frage wie wahrscheinlich es ist, dass der Proband den Brillenberater in Zukunft nutzen würde, konnte dieser die Frage auf einer Skala von 0-5 beantworten, wobei 5 für 'sehr wahrscheinlich' stand. Die durchschnittliche Bewertung bei dieser Frage lag bei 2,67 (siehe Abbildung 61). Auffallend war, dass mit einem Durchschnittswert von 2,84 die höchste Zustimmung aus der Gruppe derer kam, die sich nicht vorstellen können eine Brille online zu kaufen. Die Brillenträger erreichten einen durchschnittlichen Wert von 2,71, der nur leicht über dem der Gesamtheit lag. Die Personen, die bereits eine Brille online gekauft hatten, bewerteten im Durchschnitt mit einem Wert von 2,76.

Insgesamt ließ sich feststellen, dass der Brillenberater in der Lage ist dem Nutzer anhand seiner Gesichtsmerkmale Brillen vorzuschlagen, die dieser besser bewertet als zufällig aus-

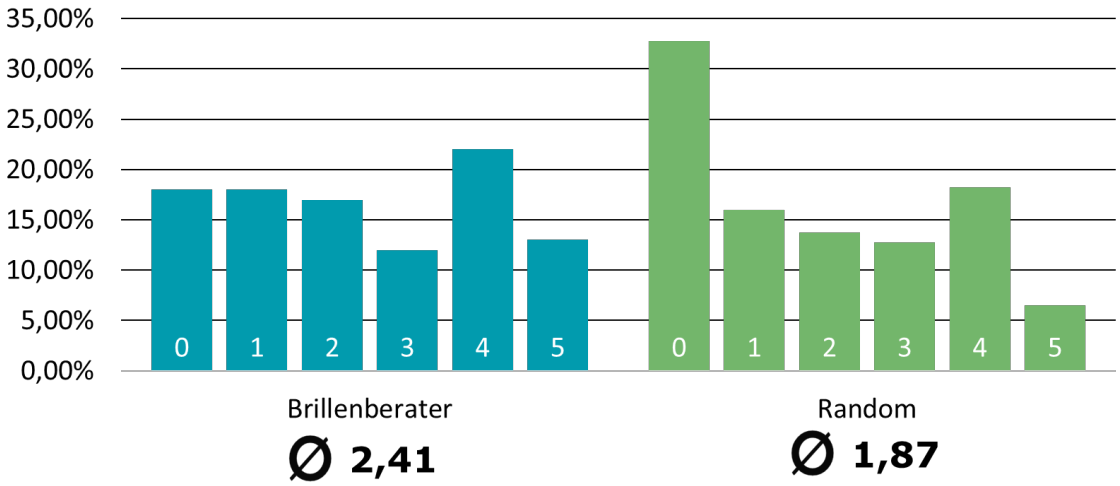


Abbildung 60: Gegenüberstellung der durchschnittlichen Bewertung der Brille, die durch den Brillenberater empfohlen wurde und der (Random-)Vergleichsbrillen (Quelle: Eigene Darstellung)

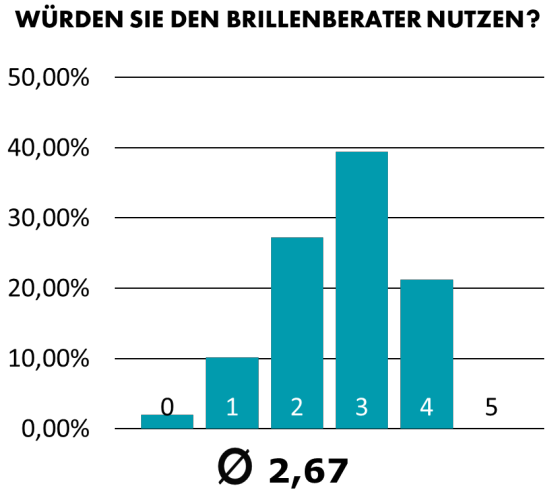


Abbildung 61: Prozentuale Verteilung der Antworten auf die Frage wie wahrscheinlich es ist, dass der Brillenberater in Zukunft genutzt werden würde (Quelle: Eigene Darstellung)

gewählte Brillen. Die Aussagekraft der Evaluation ließe sich jedoch steigern, wenn der Brillenberater mehr als die zur Verfügung stehenden 35 (bzw. ohne die Vergleichsbrillen 29) Brillen zur Auswahl gehabt hätte. Des Weiteren war die Teilnehmerzahl auch zu gering um repräsentative Ergebnisse zu erzielen. Für weitere Evaluationen wäre zudem eine Live- oder Online-Anprobe wünschenswert, da die Bewertung einer Brille stark davon abhängig ist, wie diese beim Tragen aussieht.

14 Fazit und Ausblick

Begonnen mit einem grob formulierten Konzept, der Erstellung eines intelligenten Brillenberaters auf Grundlage von neuronalen Netzen, welcher dem Kunden beim Onlinekauf einer Brille personalisierte Produktvorschläge auf Basis extrahierter Features des Gesichts generieren solle, hat sich viel im Projekt Propose.AI getan. Durch weiteres Ausbauen des Grundkonzeptes und dem kontinuierlichen Hinzufügen neuer Ideen, Anregungen und Projekten ist aus diesem grob formulierten Leitsatz im Verlauf des Projekts ein komplexes Konstrukt unterschiedlichster Services und Anwendungen entstanden.
Das Ausgangsproblem, welches Propose.AI bearbeiten sollte, war die Übertragung der persönlichen Beratung des stationären Brillenkaufs auf den E-Commerce. Der zentrale Vorteil des stationären Handels gegenüber dem Onlinehandel ist die persönliche Beratung. Kompetente Beratung vereinfacht die Suche nach der passenden Brille für den Kunden ungemein und gestaltet das Einkaufserlebnis deutlich frust- und barrierefreier. Hier sind die angebotenen Beratungsmöglichkeiten im Onlinehandel der des stationären Handels jedoch deutlich unterlegen. Wo ein Kunde von der Auswahl im stationären Handel überfordert sein könnte, bekommt dies bei der Betrachtung der Sortimentsgröße des Onlinehandels nochmal eine ganz andere Bedeutung. Aus diesem Grund müssen die Filterungsprozesse, welche die lokale Optikerin binnen Sekunden nach erstmaliger Betrachtung des Gesichts des jeweiligen Kunden durchführt, auf den E-Commerce übertragen werden, um ein ähnliches Einkaufserlebnis zu erzeugen. Auf Basis des fotografierten Gesichts des Nutzers wird durch die Extraktion spezifischer Features des Gesichts ein Filterungsprozess durchgeführt, der in einer Generierung personalisierter Produktvorschläge resultiert und somit das vorhandene Gesamtsortiment deutlich reduziert. Zur Extraktion der unterschiedlichen Features wurden neuronale Netze trainiert, welche nach dem divide-and-conquer-Prinzip auf die Extraktion einzelner Features trainiert wurden und anschließend Stück für Stück in einem Filterungsprozess abgearbeitet werden. So sind Algorithmen entstanden, welche neben der Bestimmung des Geschlechts und des Alters der jeweiligen Person auch Features wie Haar-, Hautfarbe, vorhandene Gesichtsbehaarung sowie die Gesichtsform erkennen konnten. Ein zuvor definierter Regelkatalog dient nun dazu, anhand der extrahierten Features passende Brillenvorschläge zu generieren, sodass die persönlichen Merkmale des Kunden stets Mittelpunkt der Beratung sind. Der Brillenberater war Hauptbestandteil des Projekts und wurde als einziges über den gesamten Projektverlauf bearbeitet. Zwar sind die einzelnen Modelle zur Extraktion jeweiliger Features bereits relativ robust, jedoch gibt es an der ein oder anderen Stelle auch fehlerhafte Bestimmungen, welche die Gesamtqualität der generierten Produktvorschläge verfälschen könnten. Aus diesem Grund könnte die

weitere Optimierung der Modelle Gegenstand anknüpfender Arbeiten sein. Ein zusätzlich interessanter Ansatzpunkt für weiterführende Arbeiten könnte die Anreicherung neuer Modelle zur Extraktion weiterer Features sein, welche die Funktionalität des Brillenberaters ergänzen könnten. Bisher basiert die Beratung auf rein modischen Kriterien. Durch die Erweiterung des Brillenberaters mit zusätzlichen Modellen, welche bspw. spezifische Maße des Gesichts auslesen und bestimmen, könnte der Brillenberater neben einer rein modischen Beratung auch Brillen vorschlagen, welche dem Kunden tatsächlich passen. Durch die Beschränkung der Beratung auf modische Kriterien kann es vorkommen, dass eine Brille dem Kunden zwar optisch passt, die Maße jedoch nicht zutreffen und die Brille als potentieller Produktvorschlag eigentlich ausgelassen werden sollte. Dies könnte durch die Anreicherung des Brillenberaters mit entsprechenden Modellen behoben werden.
Ein weiteres Projekt mit Kundenfokus war der Brillenpassleser. Der Brillenpassleser dient zur automatischen Erfassung der Sehwerte des Kunden. Um den Onlinekauf von Brillen noch reibungsloser zu gestalten, sollte die lästige manuelle Eingabe der Sehwerte ebenfalls von einem entsprechenden Dienst für den Kunden übernommen werden. Ziel des Brillenpasslesers ist es, mithilfe eines einfachen Fotos des Brillenpasses entsprechende Sehwerte aus dem Dokument zu extrahieren und jeweilige Input-Masken bei der Angabe der Sehwerte automatisch auszufüllen. Auf Grundlage des übermittelten Fotos werden entsprechende Felder des Brillenpasses lokalisiert und mithilfe einer Texterkennung ausgelesen, welche anschließend in das jeweilige Eingabefeld des Systems eingepflegt werden. Da der Brillenpassleser momentan ausschließlich auf Brillenpässe von Fielmann und Brille24 ausgelegt ist, kann weiterführend daran gearbeitet werden, die Erkennung auf weitere Brillenpässe auszuweiten. Ebenfalls interessant wäre ein Verfolgen des Ansatzes, auf die Übermittlung eines Fotos zu verzichten und die gewünschten Werte stattdessen in Echtzeit auszulesen, indem lediglich die Kamera über das entsprechende Dokument gehalten wird. Wie bereits erwähnt generiert der Brillenberater lediglich Produktvorschläge unter Berücksichtigung rein modischer Kriterien. Grundlage dieser Beratung sind jeweilige extrahierte Features, welche anschließend auf Basis definierter Regeln einen Filterungsprozess durchlaufen und anschließend in spezifischen Produktvorschlägen resultieren. Dies ist besonders wichtig zu erwähnen, da diese Regeln unter Berücksichtigung aktueller Modetrends erstellt wurden und in enger Zusammenarbeit der Fashion- und Modespezialisten des Praxispartners Brille24 entstanden sind. Doch was ist, wenn der Geschmack eines Nutzers diesen Modetrends widerspricht? Da eine modische Beratung rein subjektiver Natur ist, ist die Qualität bei der Beratung von Kunden, welche aktuelle Modetrends ablehnen, stark eingeschränkt. Aus dieser Erkenntnis ist der Dienst Brillentinder entstanden, welcher genau dieses Problem adressieren soll. Die Idee ist, dass Brillen mithilfe von Embeddings in eine mathemati-

sche Repräsentation umgewandelt werden, um sie so mathematisch vergleichbarer bzw. unterscheidbarer zu machen. Mithilfe intuitiver Wischgesten kann der Nutzer dem Dienst signalisieren, ob ihm eine vorgeschlagene Brille gefällt oder nicht, um so die Fortbewegung im euklidischen Raum der durch Vektoren repräsentierten Brillen zu beeinflussen und Anhand seiner Entscheidung Einfluss auf die Generierung der Brillenvorschläge zu nehmen. Durch Embeddings und deren mathematische Repräsentation können ähnliche Brillenmodelle als Cluster im euklidischen Raum abgebildet werden, wobei die mathematische Entfernung der einzelnen Objekte signalisiert, wie ähnlich bzw. unähnlich die betrachteten Objekte voneinander sind. Die Wischgesten beeinflussen nun, wie sich im euklidischen Raum fortbewegt wird, die objektiven Entscheidungen des Nutzers werden somit maßgeblich in den Prozess der Vorschlagsfindung eingebunden und der subjektive Charakter des Brillenberaters kann somit ausgeglichen werden. Da dieser Dienst erst zu einer recht späten Iterationsphase des Projektes entstanden ist, gab es keine wirkliche Möglichkeit, den Dienst effektiv zu nutzen oder einzubinden. Deshalb wäre als Ausblick die Verknüpfung der Dienste Brillenberater und Brillentinder ein interessanter Ansatzpunkt, um den modisch subjektiven Beratungscharakter mit der Objektivität des Brillentinders zu erweitern. Nach der Generierung der Produktvorschläge durch den Brillenberater könnte bspw. der Dienst des Brillentinders angeknüpft werden, um basierend auf den bisherigen Vorschlägen die Produktauswahl weiter zu verfeinern.
Der Brillenberater, Brillenpassleser sowie das Brillentinder sind Dienste, welche einen starken Kundenfokus aufweisen. Im Laufe des Projektes und aufgrund des stetigen Austausches mit dem Praxispartner Brille24 hat sich jedoch herauskristallisiert, dass ebenfalls Potenziale existieren, um auf Basis ähnlicher Machine-Learning-Verfahren intern Mehrwerte für das Unternehmen zu realisieren. Vor allem Abteilungen wie der Einkauf sind extrem auf Information angewiesen und äußerst datengetrieben. Um die Einkäufe für die nächsten Quartale zu prognostizieren, werden ausschlaggebende Daten benötigt, auf dessen Grundlage die jeweiligen Mengen entsprechender Modelle bestimmt und anschließend bestellt werden können. Dabei besteht stets das Risiko, zu viel oder zu wenig von der jeweiligen Ware bestellt zu haben. In beiden Fällen resultiert dies in entsprechenden Mehraufwänden, welche durch die Zunahme weitere Bewertungskriterien bei der Planung des Einkaufs minimiert werden könnten. Aus diesem Grundgedanken ist die Social Media Trendanalyse entstanden, mithilfe welcher aktuelle Brillentrends in sozialen Medien erkannt werden sollen, um diese Informationen anschließend in jeweilige Planungsprozesse zu integrieren. Auf Basis eines definierten Crawlingkonzepts werden Daten von ausgewählten Influencern in sozialen Medien mit Bildfokus erschlossen und in einer Datenbank abgelegt. Durch die Anwendung einer Object Detection kann sichergestellt werden, dass auf dem jeweilig erschlos-

senen Bild eine Brille abgebildet ist, um es anschließend für weitere Verarbeitungsprozesse vorzubereiten. Mithilfe verschiedener Machine-Learning-Algorithmen kann nun bestimmt werden, ob es sich bei dem abgebildeten Objekt um eine Sonnenbrille bzw. normale Brille handelt, welches Brillenmodell vorliegt, welche Rahmenart das Modell besitzt und welche Rahmenfarbe die jeweilige Brille hat. Diese Metainformationen werden nun in einer strukturierten Form an die Dashboard-Komponenten weitergegeben, welche zur Visualisierung der erschlossenen Informationen dienen. Mittels geeigneter Visualisierungsverfahren entsteht so ein Tool, welches dem Nutzer ermöglicht, interaktiv und schnell gewünschte Informationen abzurufen und zukünftige Entscheidungen auf Basis zusätzlicher Informationsgrundlagen zu tätigen. De facto stellt die Social Media Trendanalyse jedoch lediglich eine Ist-Analyse dar. Eine zusätzliche Erweiterung des Datenbestandes mit geeigneten Prognoseverfahren und -modellen könnte den Informationsgehalt des Dienstes noch weiter verstärken. Ein weiterer Ansatz wäre die Anreicherung des Datenbestandes der Social Media Trendanalyse mit den bereits vorhandenen Daten des Ein- und Verkaufs. Die Kombination tatsächlicher Absatzzahlen mit den gesammelten Metainformationen aus den sozialen Medien könnte zu weiteren interessanten Korrelationen führen.

Eingebettet werden die aufgeführten Services innerhalb der AI-Service-Plattform. Die Grundidee einer Service-Plattform ist aus der Tatsache entstanden, dass die angebotenen Services modular voneinander agieren sollten. Einzelne Komponenten der Architektur, wie bspw. die Datenbank, sollten einfach austauschbar sein, um reibungslose Migrationen auf andere Technologien zu ermöglichen. Im Mittelpunkt standen eine leichte Erweiterbarkeit sowie Skalierbarkeit, welche durch die Modularität der einzelnen Dienste und Komponenten ermöglicht werden sollte. Der Fokus auf horizontale Skalierbarkeit sollte zudem eine effektive Lastenverteilung ermöglichen und bei entsprechender Ressourcenknappheit eine einfache Anbindung neuer Hardware garantieren.

Nicht alle übergebenen Dienste sind perfekt ausgearbeitet. In manchen Bereichen, wie bspw. bei der Performance ausgewählter Netze, gibt es noch Optimierungsbedarf. Auch bei der Social Media Trendanalyse gibt es eine Vielzahl von Ansatzpunkten, um das Projekt weiter auszubauen. Insgesamt ist aber hervorzuheben, dass die anfängliche Intention der Gestaltung eines intelligenten Brillenberaters zur Generierung personalisierter Produktvorschläge auf Basis heterogener Daten und deren dazugehörig definierten Anforderungen erfolgreich realisiert werden konnte. Durch die effektive Verknüpfung der im Projekt realisierten Projekte bzw. Ergebnisse und der verwendeten Technologien mithilfe der von uns konstruierten Architektur, welche es erlaubt, jeden Dienst und jede Technologie als modulare Komponente zu betrachten, ist die Anforderung der Gestaltung einer agilen AI-Service-Plattform ebenfalls erfolgreich realisiert worden.

Literaturverzeichnis

[AC19] J. J. Allaire und François Chollet. *R Interface to 'Keras'*. 5.02.2019. URL: `https://keras.rstudio.com/articles/tutorial_overfit_underfit.html` (besucht am 23. 04. 2019).

[Agu+17] Eirikur Agustsson u. a. "Apparent and real age estimation in still images with deep residual regressors on APPA-REAL database." In: *12th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG), 2017*. IEEE. 2017.

[Alg18] Algorithmia Inc., Hrsg. *Introduction to Optimizers — Algorithmia Blog*. https://www.facebook.com/algorithmia, 2018. URL: `https://blog.algorithmia.com/introduction-to-optimizers/` (besucht am 26. 04. 2019).

[Bri18] Brille24 GmbH, Hrsg. *Factsheet mit Informationen über die Brille24 GmbH*. Oldenburg, 2018. URL: `https://www.brille24.de/presse/factsheet.html` (besucht am 30. 11. 2018).

[Bro16] Jason Brownlee. *Supervised and Unsupervised Machine Learning Algorithms*. 2016. URL: `https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/` (besucht am 23. 04. 2019).

[BT17] Adrian Bulat und Georgios Tzimiropoulos. "How far are we from solving the 2D & 3D Face Alignment problem? (and a dataset of 230, 000 3D facial landmarks)". In: *CoRR* abs/1703.07332 (2017).

[CA19a] François Chollet und J. J. Allaire. *Applications - Keras Documentation*. 11.02.2019. URL: `https://keras.io/applications/#usage-examples-for-image-classification-models` (besucht am 23. 04. 2019).

[CA19b] François Chollet und J. J. Allaire. *Convolutional Layers - Keras Documentation*. 11.02.2019. URL: `https://keras.io/layers/convolutional/` (besucht am 23. 04. 2019).

[CA19c] François Chollet und J. J. Allaire. *Core Layers - Keras Documentation*. 11.02.2019. URL: `https://keras.io/layers/core/` (besucht am 23. 04. 2019).

[CA19d] François Chollet und J. J. Allaire. *Pooling Layers - Keras Documentation*. 11.02.2019. URL: `https://keras.io/layers/pooling/` (besucht am 23. 04. 2019).

[Cho15] François Chollet. *Keras*. https://keras.io. 2015.

[Cou19] Couchdb. *10.3.6. /db/_find — Apache CouchDB 2.2 Documentation*. 2019. URL: https://docs.couchdb.org/en/2.2.0/api/database/find.html (besucht am 19. 03. 2019).

[Daw14] Kenneth Dawson-Howe. *A practical introduction to computer vision with opencv*. John Wiley & Sons, 2014.

[Dic12] Dick Hardt. *The OAuth 2.0 Authorization Framework*. 2012. URL: https://tools.ietf.org/html/rfc6749#section-1.4 (besucht am 27. 02. 2019).

[Epp11] Thomas Epping. *Kanban für die Softwareentwicklung*. de. Informatik im Fokus. OCLC: 754214825. Berlin: Springer, 2011. ISBN: 978-3-642-22594-9 978-3-642-22595-6.

[FP18a] Brendan Fortuner und Prashant Piprotar. *Activation Functions*. 2018. URL: https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html#sigmoid (besucht am 23. 04. 2019).

[FP18b] Brendan Fortuner und Prashant Piprotar. *Activation Functions*. 2018. URL: https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html#tanh (besucht am 23. 04. 2019).

[FP18c] Brendan Fortuner und Prashant Piprotar. *Activation Functions*. 2018. URL: https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html#relu (besucht am 23. 04. 2019).

[FP18d] Brendan Fortuner und Prashant Piprotar. *Activation Functions*. 2018. URL: https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html#softmax (besucht am 23. 04. 2019).

[FVK18a] Brendan Fortuner, Marcelo Viana und Bhargav Kowshik. *Loss Functions*. 2018. URL: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html# (besucht am 23. 04. 2019).

[FVK18b] Brendan Fortuner, Marcelo Viana und Bhargav Kowshik. *Loss Functions*. 2018. URL: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#cross-entropy (besucht am 23. 04. 2019).

[Glo10] Boris Gloger. "Scrum: Der Pradigmenwechsel im Projekt- und Produktmanagement – Eine Einführung". de. In: *Informatik-Spektrum* 33.2 (Apr. 2010), S. 195–200. ISSN: 0170-6012, 1432-122X. DOI: 10.1007/s00287-010-0426-6. URL: http://link.springer.com/10.1007/s00287-010-0426-6 (besucht am 01. 07. 2018).

[Gup17a] Disha Shree Gupta. *Fundamentals of Deep Learning – Activation Functions and When to Use Them?* Hrsg. von Analytics Vidhya. 2017. URL: https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning-activation-functions-when-to-use-them/ (besucht am 23. 04. 2019).

[Gup17b] Vikas Gupta. *Keras Tutorial : Using pre-trained ImageNet models — Learn OpenCV*. 2017. URL: https://www.learnopencv.com/keras-tutorial-using-pre-trained-imagenet-models/ (besucht am 23. 04. 2019).

[Ham09] Oliver-Arne Hammerstein. *Das aktuelle Sichwort: Scrum*. Apr. 2009.

[Hol12] Daniel Holth. *PEP 427 - The Wheel Binary Package Format 1.0*. Hrsg. von Nick Coghlan. 12. Sep. 2012. URL: https://www.python.org/dev/peps/pep-0427/.

[Hun07] J. D. Hunter. "Matplotlib: A 2D Graphics Environment". In: *Computing in Science Engineering* 9.3 (Mai 2007), S. 90–95.

[Int18] Internet Engineering Task Force. *JSON Schema: A Media Type for Describing JSON Documents. draft-handrews-json-schema-01*. Hrsg. von A. Wright. 19. März 2018. URL: https://json-schema.org/latest/json-schema-core.html.

[KSH12] Alex Krizhevsky, Ilya Sutskever und Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, S. 1097–1105.

[Lea+16] Erik Learned-Miller u. a. "Labeled Faces in the Wild: A Survey". In: *Advances in Face Detection and Facial Image Analysis*. Hrsg. von Michal Kawulok, M. Emre Celebi und Bogdan Smolka. Cham: Springer International Publishing, 2016, S. 189–248.

[Lin+17] Tsung-Yi Lin u. a. "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*. 2017, S. 2980–2988.

[Man17] Moritz Mannschreck. *Fast jedes vierte Unternehmen betrieb 2016 E-Commerce*. 2017. URL: https://www.destatis.de/DE/Presse/Pressemitteilungen/2017/12/PD17_446_52911.html (besucht am 09. 05. 2019).

[McK10] Wes McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Hrsg. von Stéfan van der Walt und Jarrod Millman. 2010, S. 51–56.

[Mey18] Annika Meyer. *Führender Online-Optiker steigert Umsatz durch künstliche Intelligenz. Fielmanns Schwäche ist die Zukunft von Brille24*. Hrsg. von Brille24 GmbH. 2018. URL: https://cdn.brille24.de/fileadmin/om/presse/pdf/Brille24_Pressemitteilung_KI_03.2018_final.pdf (besucht am 28. 01. 2019).

[MM12] Ondrej Miksik und Krystian Mikolajczyk. "Evaluation of local detectors and descriptors for fast feature matching". In: *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. IEEE. 2012, S. 2681–2684.

[Oli06] Travis E Oliphant. *A guide to NumPy*. Bd. 1. Trelgol Publishing, 2006.

[PSL19] Kitsuchart Pasupa, Wisuwat Sunhem und Chu Kiong Loo. "A hybrid approach to building face shape classifier for hairstyle recommender system". In: *Expert Systems with Applications* 120 (2019), S. 14–32. DOI: https://doi.org/10.1016/j.eswa.2018.11.011.

[Red+15] Joseph Redmon u. a. "You Only Look Once: Unified, Real-Time Object Detection". In: *CoRR* abs/1506.02640 (2015). arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.

[RFB15] Olaf Ronneberger, Philipp Fischer und Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, S. 234–241.

[Rie12] Wolf Riepl. *CRISP-DM: Ein Standard-Prozess-Modell für Data Mining*. 2012. URL: https://statistik-dresden.de/archives/1128 (besucht am 18. 02. 2019).

[RTG16] Rasmus Rothe, Radu Timofte und Luc Van Gool. "Deep expectation of real and apparent age from a single image without facial landmarks". In: *International Journal of Computer Vision (IJCV)* (Juli 2016).

[Rus+15a] Olga Russakovsky u. a. "Imagenet large scale visual recognition challenge". In: *International journal of computer vision* 115.3 (2015), S. 211–252.

[Rus+15b] Olga Russakovsky u. a. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), S. 211–252. DOI: 10.1007/s11263-015-0816-y.

[Sie18] Markus Siepermann. *Definition: Agile Softwareentwicklung*. de. 2018. URL: https://wirtschaftslexikon.gabler.de/definition/agile-softwareentwicklung-53460/version-276549 (besucht am 25. 06. 2018).

[Sri+14] Nitish Srivastava u. a. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (2014), S. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[Sta18] Statista. *Ranking der größten Augenoptiker in Deutschland nach Netto-Umsatz in den Jahren 2009 bis 2017 (in Millionen Euro)*. https://de.statista.com/statistik/daten/studie/206293/umfrage/groesste-augenoptiker-nach-umsatz/. [Abgerufen: 30.07.2018]. 2018.

[Sta19] Statista. *Umsatz durch E-Commerce (B2C) in Deutschland in den Jahren 1999 bis 2018 sowie eine Prognose für 2019 (in Milliarden Euro)*. 2019. URL: https://de.statista.com/statistik/daten/studie/3979/umfrage/e-commerce-umsatz-in-deutschland-seit-1999/ (besucht am 09. 05. 2019).

[Tan18] Ole Tange. *GNU Parallel 2018*. Ole Tange, Apr. 2018. DOI: 10.5281/zenodo.1146014. URL: https://doi.org/10.5281/zenodo.1146014.

[Ten19] TensorFlow. *Embeddings — TensorFlow*. TensorFlow. 2019. URL: https://www.tensorflow.org/guide/embedding (besucht am 18. 03. 2019).

[Tho19] Thomas Krenn AG. *Git Grundbegriffe – Thomas-Krenn-Wiki*. 23. Jan. 2019. URL: https://www.thomas-krenn.com/de/wiki/Git_Grundbegriffe.

[Tia+16] Zhi Tian u. a. "Detecting text in natural image with connectionist text proposal network". In: *European conference on computer vision*. Springer. 2016, S. 56–72.

[Tre15] Tobias Trepper. "Forschungsmethodik". In: *Fundierung der Konstruktion agiler Methoden*. Hrsg. von Tobias Trepper. Wiesbaden: Springer Fachmedien Wiesbaden, 2015, S. 11–28. ISBN: 978-3-658-10089-6. DOI: 10.1007/978-3-658-10090-2_2.

[Uni18] University of Masschusetts, Hrsg. *Labeled Faces in the Wild*. Masschusetts, 2018. URL: http://vis-www.cs.umass.edu/lfw/ (besucht am 18. 02. 2019).

[VV15] Kethineni Venkateswarlu und Sreerama Murthy Velaga. "Text detection on scene images using MSER". In: *International Journal of Research in Computer And Communication Technology* 4.7 (2015), S. 452–456.

[WJ19] Yue Wu und Qiang Ji. "Facial Landmark Detection: A Literature Survey". In: *International Journal of Computer Vision* 127.2 (Feb. 2019), S. 115–142.

[WM08] H.W. Wieczorrek und P. Mertens. *Management von IT-Projekten: Von der Planung zur Realisierung*. Xpert.press. Springer Berlin Heidelberg, 2008. ISBN: 9783540852902. URL: https://books.google.de/books?id=FBnZVpUMJzMC.

[Zel17] H. Zell. *Projektmanagement: - zehn Module zu ausgewählten Themen*. Books on Demand, 2017. ISBN: 9783744876865. URL: https://books.google.de/books?id=xTTKDgAAQBAJ.

[ZSQ17] Zhifei Zhang, Yang Song und Hairong Qi. "Age Progression/Regression by Conditional Adversarial Autoencoder". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017.

A Anhang

A.1 Schema für externe Anfrage (ServiceRequest)

```
{
  "$id": "https://pg.joelt.de/schema/v1/service-request.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "ServiceRequest",
  "type": "object",
  "properties": {
    "serviceName": {
      "type": "string",
      "description": "The name of the service that is registered."
    },
    "apiVersion": {
      "type": "string",
      "description": ""
    },
    "requestPayload": {
      "type": "object",
      "description": "Request payload that will be parsed by the backend
    service."
    },
    "modelVersion": {
      "type": "string",
      "description": "Model version to be used by the service."
    }
  },
  "required": [
    "serviceName",
    "apiVersion",
    "requestPayload"
  ]
}
```

Architektur/listings/service_request_external_schema.json

A.2 Schema für interne Anfrage (ServiceRequestInternal)

```
{
  "$id": "https://pg.joelt.de/schema/v1/service-request-internal.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "ServiceRequestInternal",
  "type": "object",
  "properties": {
    "requestPayload": {
      "type": "object",
      "description": "Request payload that will be parsed by the backend service."
    },
    "requestId": {
      "type": "string",
      "description": "ID of the request for tracking purposes."
    },
    "modelVersion": {
      "type": "string",
      "description": "Model version to be used by the service."
    }
  },
  "required": [
    "requestPayload",
    "requestId"
  ]
}
```

Architektur/listings/service_request_internal_schema.json

A.3 Schema für interne Antwort (ServiceResponseInternal)

```
{
  "$id": "https://pg.joelt.de/schema/v1/service-response-internal.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "ServiceResponseInternal",
  "type": "object",
  "properties": {
    "responsePayload": {
      "type": "object",
      "description": "Response payload that will be parsed by the backend service."
    },
    "requestId": {
```

```
12        "type": "string",
13        "description": "ID of the request for tracking purposes."
14      },
15      "responseStatus": {
16        "type": "string",
17        "description": "Type of the response message. ",
18        "enum": [
19          "ok",
20          "error"
21        ]
22      },
23      "modelVersion": {
24        "type": "string",
25        "description": "Model version to be used by the service."
26      },
27      "serviceName": {
28        "type": "string",
29        "description": "Name (type9 of the service that processed the request
        ."
30      },
31      "serviceInstanceId": {
32        "type": "string",
33        "description": "ID of the worker that processes the request."
34      },
35      "apiVersion": {
36        "type": "string",
37        "description": "API Version required to answer this request."
38      },
39      "startedAt": {
40        "type": "string",
41        "description": "ISO timestamp of the processing begin."
42      },
43      "finishedAt": {
44        "type": "string",
45        "description": "ISO timestamp for the end of the processing."
46      }
47    },
48    "required": [
49      "responsePayload",
50      "requestId",
51      "serviceInstanceId",
52      "serviceName",
53      "apiVersion",
54      "startedAt",
```

```
55      "finishedAt",
56      "responseStatus"
57    ]
58  }
```

Architektur/listings/service_response_internal_schema.json

A.4 setup.py der Basisbibliothek

```
from setuptools import setup, find_packages
import subprocess

VERSION_INFO = ('0', '1', '0', 'dev0')

def __extract_git_revision():
    """
    Extract the short revision id of the current HEAD.

    Returns:
        str: Short revision of the current git HEAD
    """
    return (subprocess.check_output('git rev-parse --short HEAD', shell=True
    )).decode('ascii').replace('\n', '')

def __update_version():
    with open('paiservice/version.py', 'w') as f:
        f.write("""# AUTO GENERATED - DO NOT EDIT
__version__ = '{version}'
__version_info__ = {version_info}
__revision__ = '{revision}'
""".format(version='.'.join(VERSION_INFO),
           version_info=str(VERSION_INFO),
           revision=__extract_git_revision()))
        return '.'.join(VERSION_INFO)

setup(
    name='paiservice',
    version=__update_version(),
    url='https://pg.joelt.de',
    license='',
    author='Jannis Oeltjen',
    author_email='oss@jcoeltjen.de',
    description='Propose.AI Base Library',
    package_data={'paiservice.schema': ['*.json']},
    include_package_data=True,
    install_requires=[
        'PyYAML==3.13',
        'redis==2.10.6',
        'jsonschema==2.6.0',
        'petname==2.2'
    ],
    setup_requires=[
        'pytest-runner',
        'wheel'
    ],
    tests_require=[
        'pytest',
        'pytest-cov'
    ],
    packages=find_packages(exclude=['tests', 'tests.*']),
    entry_points={
        'console_scripts': ['paiservice=paiservice.commandline.
    cli_entrypoint:main']}
```

A.5 Testergebnisse der Lasttests

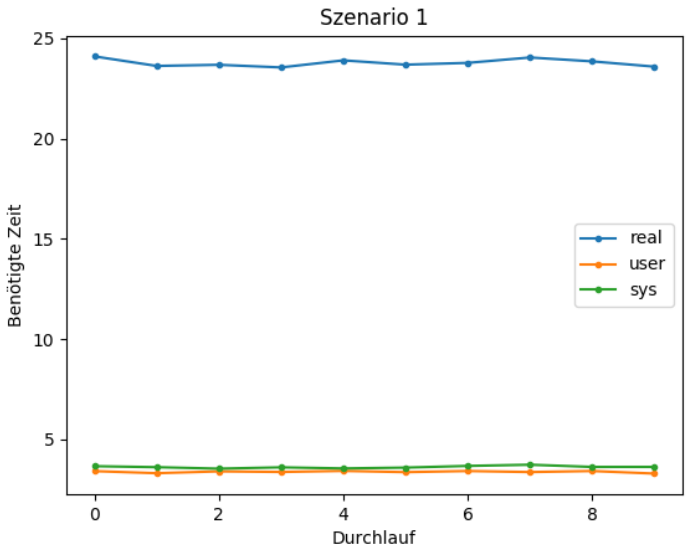


Abbildung 62: Testergebnisse Szenario 1

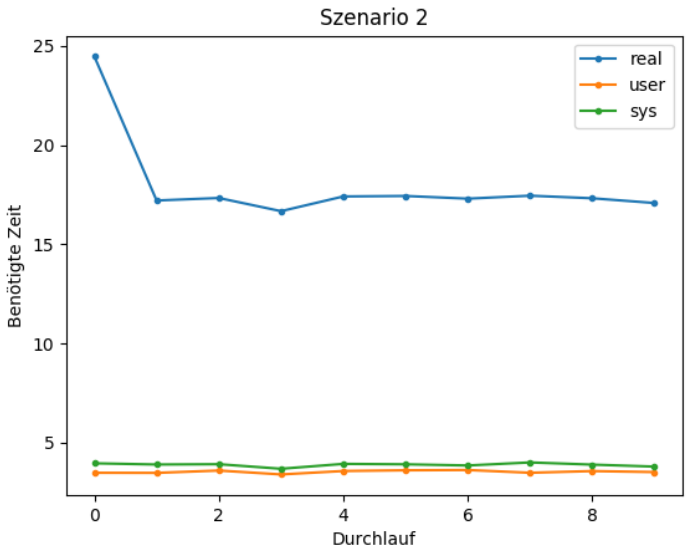


Abbildung 63: Testergebnisse Szenario 2

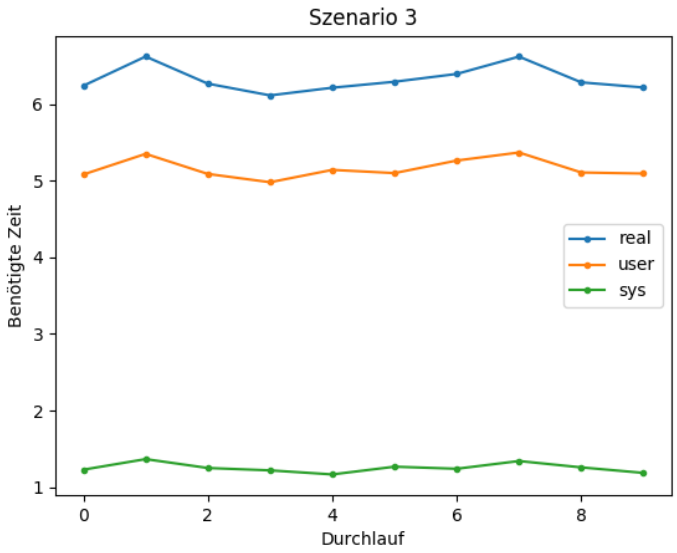


Abbildung 64: Testergebnisse Szenario 3

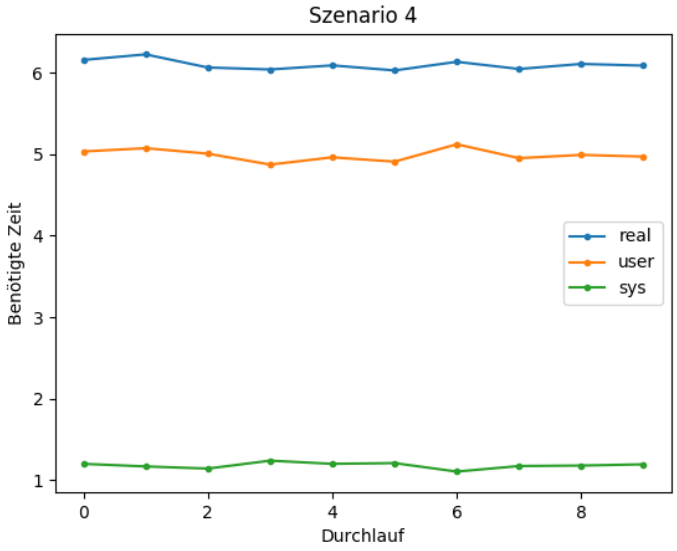


Abbildung 65: Testergebnisse Szenario 4

A.6 Projektpläne

A.6.1 Projektplan Phase 2

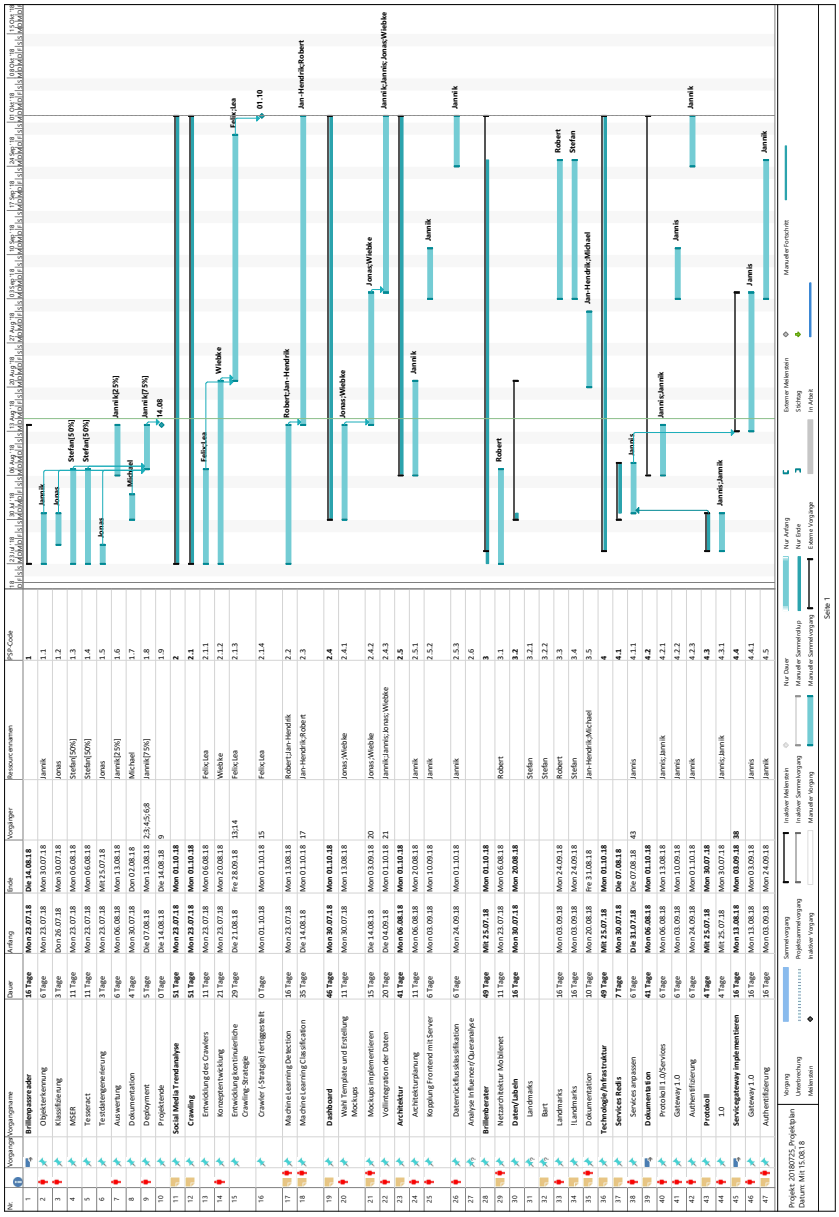


Abbildung 66: Gantt Diagramm für die zweite Projektphase [Eigene Darstellung]

A.6.2 Projektplan Phase 3 u. 4

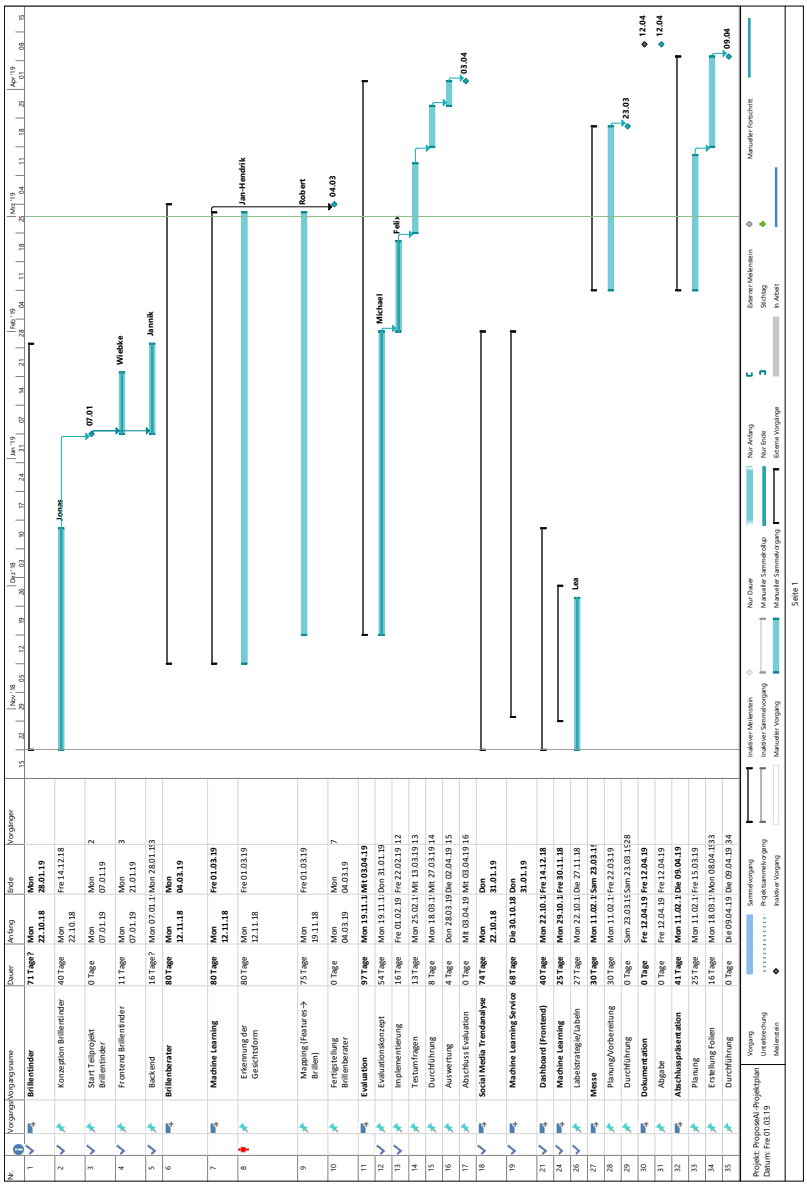


Abbildung 67: Gantt Diagramm für die dritte und vierte Projektphase [Eigene Darstellung]

A.7 Evaluation Brillenberater - Fragebogen erster Entwurf

Nr.	Fragestellung	Fragetyp	Antwortmöglichkeiten
1	Geschlecht	Einfachnennung	Männlich | Weiblich | Keine Angabe
2	Alter	Einfachnennung	Altersgruppen: 14-25 | 26-39 | 40-55 | Älter als 55
3	Brillenträger	Einfachnennung	Ja | Nein
4	Wie häufig kaufen Sie online?	Einfachnennung	Mehrmals Pro Monat | Einmal pro Monat | Einmal im Quartal | Einmal im halben Jahr | Einmal im Jahr | Seltener als einmal im Jahr | nie
5	Haben Sie schon eine Brille im Internet gekauft?	Einfachnennung	Ja | Nein
	Wenn Frage 5 mit Ja beantwortet wurde:		
6	Um welche Art von Brillen handelte es sich bei Ihren Brillenkäufen im Internet?	Einfachauswahl	Sonnen- und Korrektionsbrillen | Korrektionsbrillen | Sonnenbrillen | Keine Angabe
7	Wie zufrieden waren Sie mit ihrer letzten online gekauften Brille?	Likert-Item [0-5]	Unzufrieden - Zufrieden
8	Bei welchem der folgenden Online-Optiker haben Sie ihren letzten Brillenkauf im Internet durchgeführt?	Einfachauswahl	Mister Spex | Brille24 | Apollo | eyes + more | myspexx.de | edel-optics.de | SmartBuyGlasses | Sonstiger Anbieter
9	Wie wahrscheinlich ist es, dass Sie noch einmal eine Brille im Internet kaufen?	Likert-Item [0-5]	Unwahrscheinlich - Wahrscheinlich
	Wenn Frage 5 mit Nein beantwortet wurde:		
10	Könnten Sie sich vorstellen eine Brille im Internet zu kaufen?	Einfachauswahl	Ja, Sonnen- und Korrektionsbrillen | Ja, Sonnenbrillen | Ja, Korrektionsbrillen | Nein

Nr.	Fragestellung	Fragetyp	Antwortmöglichkeiten
11	Welche der folgenden Online-Optiker kennen Sie?	Mehrfachauswahl	Mister Spex | Brille24 | Apollo | eyes + more | myspexx.de | edel-optics.de | SmartBuyGlasses | Sonstiger Anbieter
12	Welche der folgenden Aussagen treffen für Sie auf Filialoptiker zu?	Likert-Skala [0-5]	Trifft voll und ganz zu - Trifft gar nicht zu
12.1	Bietet mir eine gute Beratung	Likert-Item	s. o.
12.2	Bietet eine große Auswahl	Likert-Item	s. o.
12.3	Ist kompetent	Likert-Item	s. o.
12.4	Ist günstig	Likert-Item	s. o.
12.5	Sorgt dafür, dass ich als Kunde zufrieden bin	Likert-Item	s. o.
13	Welche der folgenden Aussagen treffen für Sie auf Online-Optiker zu?	Likert-Skala [0-5]	Trifft voll und ganz zu - Trifft gar nicht zu
13.1	Bietet mir eine gute Beratung	Likert-Item	s. o.
13.2	Bietet eine große Auswahl	Likert-Item	s. o.
13.3	Ist kompetent	Likert-Item	s. o.
13.4	Ist günstig	Likert-Item	s. o.
13.5	Sorgt dafür, dass ich als Kunde zufrieden bin	Likert-Item	s. o.
14	Wenn Sie an ihren nächsten Brillenkauf denken, werden Sie diesen bei einem Online- oder Filialoptiker durchführen?	Einfachauswahl	Filiale | Online | Ich bin mir nicht sicher
Vorstellung Brillenberater (Erklärung, Live-Test oder Video)			

Nr.	Fragestellung	Fragetyp	Antwortmöglichkeiten
15	Angenommen Sie wollen eine Brille im Internet kaufen. Wie wahrscheinlich wäre es, dass Sie den Ihnen gerade vorgestellten Brillenberater nutzen würden?	Likert-Item [0-5]	Unwahrscheinlich - Wahrscheinlich
16	Angenommen Sie stehen vor der Entscheidung eine Brille online oder in einer Filiale zu kaufen. Wäre der Brillenberater für Sie ein entscheidendes Argument um ihre Brille online zu kaufen?	Likert-Item [0-5]	Trifft voll und ganz zu - Trifft gar nicht zu
17	Feedback	—	Freitext

Tabelle 31: Fragebogen

A.8 Seminararbeiten

Convolutional Neural Networks – Feature Extractor

Robert Albrecht[1]

Abstract: Convolutional Neural Networks (ConvNets) stellen heutzutage den State of the Art in Bezug auf Computer Vision dar. Erstmals aufgekommen ist diese Art der neuronalen Netze Ende der 1980er Jahre [Le89]. Ihre prädominante Stellung der letzten Jahre jedoch haben sie vor allem den rekordbrechenden Performances bei Wettbewerben wie den ImageNet Large Scale Image Recognition Challenges zu verdanken, angefangen bei dem mittlerweile unter der Bezeichnung AlexNet geläufigen ConvNet von Krizhevsky et al. aus dem Jahr 2012 [KSH12]. Eine Eigenschaft, die ConvNets so mächtig macht, ist ihre Fähigkeit, Feature zu extrahieren. Der Fokus dieser Seminararbeit soll deshalb darauf liegen, diese näher zu erläutern. Hierbei sollen zunächst die Grundlagen von ConvNets erklärt werden. Der Fokus hierbei liegt auf den Convolutional und Pooling Layern, da diese für die Feature Extraction verantwortlich sind. Anschließend werden am Beispiel des bereits erwähnten Papers [KSH12] der Aufbau eines beispielhaften ConvNets und die Relevanz von ImageNet erklärt. Basierend auf diesen Erkenntnissen soll dann anhand verschiedener Paper aufgezeigt werden, wie trainierte ConvNets als Feature Extractor genutzt werden können. An dieser Stelle setzt dann der letzte Abschnitt an, der sich dem Thema widmet, inwieweit die Relevanz des Themas für PROPOSE.AI gegeben ist und wie die Feature Extraction genutzt werden könnte.

1 Grundlagen

Zunächst soll der Aufbau von ConvNets im Vergleich zu gewöhnlichen neuronalen Netzen beschrieben werden. Der Convolution Layer sowie der Pooling Layer der ConvNets sollen hierbei detaillierter erläutert werden, da in diesen Schichten die Feature Extraction stattfindet.

Sowohl bei ConvNets als auch bei gewöhnlichen neuronalen Netzen stellt der Input die erste Schicht dar. An dieser Stelle unterscheiden sich die Arten jedoch bereits: Angenommen der Input ist ein Farbbild.[2] Ein solches kann als dreidimensionale Matrix interpretiert werden mit je einer Dimension für die Höhe und Breite, wobei jeweils die Anzahl der Pixel relevant ist, und einer zusätzliche Dimension, der Tiefe, für die drei Farbkanäle (rot, blau, gelb). Ein ConvNet kann diese dreidimensionale Matrix als Input verarbeiten, bei einem gewöhnlichen neuronalen Netz hingegen müsste die Matrix erst in einen Spaltenvektor transformiert werden. [Ka18]

[1] Carl von Ossietzky Universität Oldenburg, Very Large Business Applications, Ammerländer Heerstraße 114, 26129 Oldenburg, Deutschland, robert.albrecht@uni-oldenburg.de

[2] Dieser Anwendungsfall ist für die Projektgruppe besonders relevant, da etwa für den Brillenpassreader und auch für den Brillenberater Fotos verarbeitet werden müssen.

Bei den gewöhnlichen neuralen Netzen folgen auf den Input einer oder mehrere Hidden Layer, sofern es sich nicht um ein Single Layer Perceptron handelt. Hierbei sind zwischen benachbarten Schichten jeweils alle Neuronen miteinander verbunden. Bei den ConvNets hingegen schließt zunächst ein Convolutional Layer an den Input an. Diese Schicht – wie das ConvNet als Ganzes – verdient der linearen mathematischen Operation der Convolution respektive Faltung ihren Namen. An dieser Stelle sollen nicht die genauen mathematischen Details der Faltung erläutert werden, da dies den Rahmen dieser Seminararbeit sprengen würde.[3] Vielmehr interessant ist, wie die Faltung bei den ConvNets funktioniert: Von zentraler Bedeutung sind die Filter.[4] Diese sind zweidimensionale quadratische Matritzen, deren Elemente die Gewichte darstellen. Das Besondere hierbei ist, dass die Filter kleiner als der Input sind. Dies bedeutet konkret, dass ein Output-Neuron immer nur mit einem kleinen lokalen Bereich[5] des vorangegangen Layers verbunden ist, im Gegensatz zu Multi Layer Perceptrons, bei denen die Neuronen „fully connected“ sind. Ein Output-Neuron wird als Summe der elementweisen Multiplikation von Input und Gewicht bestimmt. Anschließend wird der Filter zunächst entlang der Breite und anschließend entlang der Höhe weiter verschoben, um so verschiedene Output-Neuronen zu erhalten. Das Ergebnis dieser Operation ist eine zweidimensionale Matrix, die Feature Map genannt wird. Die zwei Dimensionen kommen daher zustande, dass gleichzeitig Filter auf allen 3 Farbkanälen angewendet werden, die Ergebnisse werden dann anschließend summiert, sodass die dritte Dimension entfällt. Für den Convolutional Layer können verschiedene Hyperparameter spezifiziert werden, welche entscheiden, wie viele Output-Neuronen entstehen: Die Tiefe des Outputs (T), Stride (S), Zero-Padding (P) sowie die Größe des Filters (F). Die Tiefe des Outputs wird durch die Anzahl der Filter festgelegt. Stride beschreibt die Schrittlänge, also um wie viele Pixel der Filter verschoben wird. Beim Zero-Padding werden am Rand des Inputs Nullen hinzugefügt, was hilfreich sein kann, um die Größe des Outputs zu kontrollieren. Die Größe des Filters legt das Receiptive Field eines Neurons im Convolutional Layer fest und somit auch die Anzahl an Gewichten pro Neuron. Anhand der Hyperparameter und der Größe des Inputs (W) lässt sich die Größe des Outputs direkt kalkulieren: $(W - F + 2P)/S + 1$. Bei einem 32x32 Input mit 10x10 Filter, Stride 2 und ohne Zero-Padding, also $(32 - 10 + 2 \cdot 0)/2 + 1 = 12$, wäre der Ouput somit 12x12x1, sofern lediglich ein Filter verwendet wird. [GBC16, Ka18, RW17]

Die darauffolgende Schicht ist der ReLU-Layer. Innerhalb dieser wird eine nichtlineare Aktivierungsfunktion, die Rectified Linear Unit, auf die einzelnen Elemente angewandt. Diese hat die folgende Form: $f(x) = max(0, x)$. Es werden also negative Elemente gleich 0 gesetzt, positive Werte bleiben unverändert. Im Gegensatz zu den früher etablierten Aktivierungsfunktionen wie Sigmoid oder der Tangens-hyperbolicus-Funktion handelt es sich bei ReLU um eine nicht beschränkte Funktion. Sie ist somit nicht vom Vanishing Gradient Problem [BSF94] betroffen. [GBC16, Ka18]

[3] Zudem entspricht die Faltung bei den ConvNets auch nicht strikt der mathematischen Definition, sondern lässt sich als Kreuzkorrelation betrachten. Bei der Faltung würden nämlich die Filter vor der Anwendung vertikal und horizontal vertauscht werden. [GBC16]

[4] In der Fachliteratur sind die Filter auch unter den Bezeichnungen Kernel oder Feature Detector geläufig.

[5] Das sogenannte Receiptive Field.

Der Pooling-Layer stellt die nächste Schicht dar. Innerhalb dieser Schicht findet ein Subsampling entlang der Dimensionen von Breite und Höhe statt, sodass die Anzahl der Neuronen reduziert wird. Auf diese Weise soll einerseits Overfitting verhindert werden, andererseits wirkt sich die geringe Anzahl an Parametern positiv auf die Performance aus. Am weitesten verbreitet ist das 2x2-Max-Pooling. Bei diesem wird in jedem 2x2 Bereich innerhalb einer Feature Map lediglich der maximale Wert genommen. Die Anzahl der Parameter reduziert sich auf diese Weise um 75 Prozent. Dadurch, dass nur das maximale Element innerhalb eines bestimmten Bereiches berücksichtigt wird, kann zudem eine Invarianz gegenüber kleineren Translationen (Rotationen, Verschiebungen, unterschiedliche Skalierung etc.) des Inputs erreicht werden. [GBC16, RW17, Ka18]

Die drei zulezt vorgestellten Schichten sind zusammen für die Feature Extraction zuständig. Um ein tiefes ConvNet zu erzeugen, werden mehrere der vorgestellten Layer hintereinander verwendet, auf einen Convolutional Layer (mit ReLU-Aktivierung) kann ein weiterer Convolutional Layer oder auch ein Pooling Layer folgen.[6] Äquivalent zu den Multi Layer Perceptrons gibt es am Ende einen fully-connected-Layer, der anhand einer nichtlinearen Aktivierungsfunktion (Softmax, SVM etc.) die Scores der Outputklassen besimmt und somit für die Klassifzierung zuständig ist. Ebenfalls äquivalent zu gewöhnlichen neuronalen Netzen können dann anhand von Loss-Function, Gradientenabstiegsverfahren und Backpropagation die Parameter respektive Gewichte trainiert werden. [GBC16, RW17]

2 Convolutional Neural Networks für ImageNet

ImageNet [De09] stellt eine Datenbank von über 15 Millionen gelabelten, hochauflösenden Bildern aus über 22000 Kategorien dar, die in der Forschung genutzt wird. Den Grundstein für die weite Verbreitung von ConvNets im Bereich der Computer Vision legten Krizhevsky et al. Mit ihrer ConvNet-Architektur [KSH12] erzielten sie rekordverdächtige Ergebnisse bei den ImageNet-Large-Scale-Visual-Recognition-Challenge-Wettbewerben (ILSCVRC[7]) der Jahre 2010 sowie 2012 und erzielten damit einen neuen State of the Art. Die Wettbewerbe beschäftigen sich mit der Thematik der Objektklassifizierung, das heißt die ConvNet-Architektur wird dafür genutzt, die Kategorie der Bilder aus dem Testset zu bestimmen und diese dementsprechend korrekt zu labeln. Bei dem ILSVRC-2012-Wettbewerb z.B. erzielten sie eine Top-5-Fehlerrate[8] von 15,3 Prozent, der zweitbeste Beitrag dahingegen lag bei 26,2 Prozent. In ihrem Paper stellen die Autoren ihr ConvNet vor und vergleichen diese mit gewöhnlichen feedforward-Netzen. Das sogenannte AlexNet verarbeitet Inputs

[6] Mitunter finden mittlerweile auch Architekturen Anwendung, in denen gänzlich auf Pooling Layer verzichtet wird. [Sp14]

[7] Bei den Wettbewerben wird eine Teilmenge der ImageNet-Datenbank genutzt mit zirka 1000 Bildern in 1000 Kategorien; insgesamt gibt es ungefähr 1,2 Millionen Trainingsbilder, 50000 zur Validierung und 150000 Testbilder

[8] Es wird geprüft, ob die tatsächlich korrekte Klasse in den Top 5 Klassen des Outputs ist.

der Größe 224x224x3 mit Stride 4 und ohne Zero-Padding.[9] Es werden insgesamt 5 Convolutional Layer mit ReLU-Aktivierung, 3 Max-pooling-Layer sowie 3 fully-connected Layer verwendet, wobei der letzte mit einem 1000-Wege-Softmax-Klassifizierer verbunden ist. Des Weiteren benutzen die Autoren zwei differente Verfahren zur Prävention von Overfitting. Zum einen wird das Datenset künstlich vergrößert (Data Augmentation), indem 224x224 (respektive 227x227) Pixelbereiche anstatt des ursprünglichen 256x256 Inputs genommen werden. Zudem verwenden die Autoren das sogenannte Dropout-Verfahren [Sr14]. Hierbei wird der Output eines jeden Neurons im Hidden Layer mit einer gewissen Wahrscheinlichkeit, in diesem Fall 50 Prozent, auf Null gesetzt. Dadurch wird ausgehend von diesem Neuron nicht vorwärts propagiert, sodass es dementsprechend auch nicht an der Backpropagtion beteiligt ist. Ein Neuron kann dank Dropout robustere Feature lernen, da Co-Adaptionen reduziert werden. In ihrem Paper kommen Alex et al. zu dem Ergebnis, dass ConvNets im Vergleich zu gewöhnlichen neuronalen Netzen mit ähnlich vielen Schichten sehr viel weniger Verbindungen und Parameter besitzen und deswegen einfacher zu trainieren sind. Dabei sei ihre theoretisch[10] im Verhältnis wahrscheinlich nur wenig schlechter. Zudem merken sie an, dass die Performance ihres Netzes merkbar schlechter wurde, wenn sie auch nur einen der Layer entfernten.

Die rekordebrechenden Ergebnisse von Krizhevsky et al. führten zu einem neuen State of the Art. In den darauffolgenden Jahren sollten ConvNets das Zentrum vieler Forschungsarbeiten sein, die sich genauer mit den Netzen auseinandersetzten. Einige von ihnen verbesserten den State of the Art weiter, indem etwa die Hyperparametern weiter feinjustiert wurden, andere versuchten zu visualisieren, was in den ConvNets passiert und wieder andere widmeten sich dem Thema der Feature Extraction und untersuchten, inwieweit vortrainierte Netze (aus)genutzt werden können. Einige solcher Paper sollen im nächsten Abschnitt vorstellt werden. Der Grund, weshalb ein Abschnitt dem Thema ImageNet gewidmet ist, ist der, dass ImageNet eine große Menge an gelabelten Datenmengen bereitstellt und in Bezug auf die Feature Extraction häufig Architekturen entwickelt und erforscht wurden, deren generischer Feature Extraktor aus einem mit ImageNet vortrainierten ConvNet stammt.

3 Feature Extraction und Transfer Learning

Allgemein können drei verschiedene Arten zum Aufbau einer ConvNet-Architektur unterschieden werden. Die erste Option besteht darin, ein ConvNet von Grund auf neu zu trainieren. Grundlage der zweiten und dritten Möglichkeit ist es, ein vortrainiertes Netz zu benutzen und den Output eines Layers als Feature Extractor zu verwenden. Dieser Feature Extractor kann dann entweder konstant gelassen werden, d.h. man lässt die Paramter

[9] [Ka18] weist darauf hin, dass der Input unmöglich 224x224 Pixel groß sein kann, sondern 227x227 sein müsste. Dies kann mit der bereits vorgestellten Formel erklärt werden. Krizhevsky et al. erhalten als Output des ersten Convolutional Layers eine Feature Map der Größe 55x55x96. Das Ergebnis der Formel müsste nach Einsetzen also 55 sein, dies ist nur der Fall, wenn der Input 227 Pixel beträgt, bei 224 geht die Formel hingegen nicht auf.

[10] Die praktisch beste Performance ist logischerweise durch die Anzahl der gelabelten Daten, die GPU usw. beschränkt.

unverändert, und es wird lediglich ein Klassifizierer mit SVM oder Softmax trainiert oder man betreibt Finetuning und passt die Gewichte wiederum per Backpropagation an die neue Aufgabe an. [Ka18] In der Folge sollen die Erkenntnisse aus verschiedenen Papern zusammengefasst werden, die sich mit der Feature Extraktion beschäftigt haben.

In [ZF14] wird eine Methode vorgeschlagen, um ConvNets zu visualisieren. Mit Hilfe dieser Methode können Zeiler und Fergus dann einige Aussagen über die Feature treffen, die extrahiert werden. Demnach seien die Feature keine zufälligen und nicht interpretierbaren Muster. Es zeigt sich, dass ein ConvNet mächtigere Feature lernt, je tiefer man die Architektur durchdringt. Zudem zeigen sich einige wünschenswerte Eigenschaften mit zunehmender Tiefe der Schichten wie steigende Invarianz. Die Autoren verglichen zudem ihre genutzte Architektur mit einem ConvNet, welches sie neu trainierten. Es zeigten sich schlechte Ergebnisse bei dem neu trainierten Netz. Dies zeige, dass es nicht sinnvoll sei, ein großes ConvNet mit einer geringen Datenmenge zu trainieren. Demgegenüber merken sie an, dass generische Feature Extractor, die auf ImageNet basieren, gute Ergebnisse bei anderen Datensets erzielen (unter Betrachtung der gleichen Aufgabe der Objektklassifizierung).

Als Basis von [Ra14] dient das ConvNet Overfeat. Dieses wurde ursprünglich zur Objektklassifikation im Rahmen von ILSVRC13 trainiert. Die aus Overfeat extrahierten Feature werden als generische Bildrepräsentation genutzt, die dann auf verschiedene Recognition Tasks (Attributdetektion uvm.) angewandt wird. Dabei ist es so, dass die Aufgaben und auch die Datensets graduell weiter vom Original abrücken. Die Autoren extrahierten Featurevektoren aus jeder einzelnen Schicht von OverFeat, um diese vergleichen zu können. Zur Klassifizierung für die neue Aufgabe wurde jeweils eine lineare SVM genutzt. Razavian et al. kommen zu dem Ergebnis, dass die Performance kontinuierlich besser wurde (mit Ausnahme der letzten beiden fully-connected-Layer, bei denen die Performance annähernd konstant war), je tiefer die Schicht in der Architektur war, aus der extrahiert wurde. Die Autoren vermelden mehrere State-of-the-Art-Ergebnisse, betonen aber dennoch die Wichtigkeit, die ConvNet-Architektur an die jeweilige Aufgabe anzupassen, sofern es die Ressourcen ermöglichen. Allgemein kommen sie zu dem Fazit, dass generische Deskriptoren, die aus ConvNets extrahiert wurden, sehr mächtig seien.

In [Yo14] untersuchen die Autoren, wie transferierbar Features aus tiefen neuronalen Netzen (in diesem Fall ConvNets) sind. Zu Beginn merken Yosinski et al. an, dass ConvNets in den ersten Schichten Feature lernen, die Gabor-Filtern oder einfachen Farbklecksen ähneln. Diese scheinen nicht spezifisch für eine bestimmte Aufgabe (Image Classficiation, Object Detection etc.) oder ein bestimmtes Datenset zu sein, sondern vielmehr allgemein anwendbar auf unterschiedliche Aufgaben/Datensets. Die Autoren setzen es sich zur Aufgabe, die Transition der allgemeinen Feature der ersten Schichten zu den aufgaben-/ und datenspezifischen Features der tiefen Layer zu untersuchen. Dabei wollen sie herausfinden, inwieweit sich die Feature für Transfer Learning nutzen lassen. Basis der Untersuchungen stellt ein ConvNet aus 8 Schichten dar, welches mit ImageNet trainiert wurde. Es werden nun jeweils n Schichten des Basisnetzes genommen mit $n \in \{1, \ldots, 7\}$ und die verbleibenden

$8 - n$ Schichten des Zielnetzes werden dann für die neue Aufgabe trainiert.[11] Das Transfer Learning im Paper wird dabei einmal mit und einmal ohne Fine-Tuning der extrahierten Feature betrieben. Hierzu merken Yosinski et al. noch an, dass die Wahl, ob Fine-Tuning betrieben wird, von der Größe des Ziel-Datensets und der Anzahl der Parameter in den ersten n Schichten abhängen sollte. Bei kleiner Datenmenge und großer Anzahl an Parametern besteht die Gefahr des Overfitting bei Fine-Tuning. Bei umgedrehten Verhältnissen, also großer[12] Datenmenge und verhältnismäßig wenigen Parametern, besteht diese Gefahr nicht und Fine-Tuning kann zu besserer Performance führen. Die Autoren gelangten zu mehreren interessanten Ergebnissen: Die Transferierbarkeit der Feature wird durch zwei Aspekte negativ beeinflusst. Der erste ist die zu erwartende Spezialisierung der Feature aus den tieferen Schichten auf die originale Aufgabe. Der Einfluss dieses Aspekts erscheint monoton, d.h. je tiefer die Schicht, desto schlechter die Performance. Der zweite Effekt, den Yosinski et al. nicht erwarteten, fand bei den mittleren Schichten statt. Hier führte der Split aufgrund der fragilen Co-Adaption zwischen den Layern zu einer schlechteren Performance. Des Weiteren stellten die Autoren fest, dass die Transferierbarkeit von extrahierten Features schlechter wird, je weiter die Zielaufgabe von der Originalaufgabe entfernt ist. Allerdings, so eine weitere Erkenntnis, seien transferierte Features in jedem Fall besser im Vergleich zu zufällig initialisierten Gewichten. Das letzte Ergebnis hat für die Projektgruppe die wahrscheinlich höchste Relevanz: Es zeigte sich, dass Fine-Tuning der transferierten Feature bezogen auf die neue Aufgabe zu einem Performance-Anstieg führte. Dabei war es egal, aus welcher Schicht die Feature extrahiert wurden. Der Effekt, dass das ConvNet das ursprüngliche Datenset gesehen hat, blieb auch nach dem Finetuning bestehen, obwohl sich Original- und Zieldatenset (deutlich) unterschieden.

In [SZ14] untersuchen Simonyan und Zisserman den Effekt der Tiefe eines ConvNets auf die Genauigkeit in Bezug auf Large Scale Image Recognition. Dabei erzielten sie State-of-the-Art-Ergebnisse (u.a. ImageNet 2014) mit ihrer Archtiektur, welche auf kleine Filter (3x3), dafür jedoch bedeutend mehr[13] Schichten mit Gewichten (d.h. Convolutional- und fully-connected-Layer) setzt im Vergleich zu etablierten Netzen wie dem bereits vorgestellten AlexNet. Die Autoren gelangen zu dem Ergebnis, dass tiefe Netze mit kleinen Filtern flache Netze mit großen Filtern übertreffen (bei vergleichbarer Gesamtzahl an Gewichten). Innerhalb des Artikels evaluieren Simonyan und Zisserman zudem, wie sich ihr ConvNet, welches mit ILSVRC vortrainiert wurde, als Feature Extractor für andere, kleinere Datensets eignet, bei denen, sofern von Grund auf trainiert, die Gefahr des Overfitting bestünde. Hierfür entfernten sie den letzten fuly-connected-Layer aus ihrer Architektur, der für die 1000-Wege-ILSVRC-Klassifikation zuständig war. Die Autoren nutzten dann die 4096-dimensionalen Aktivierungen der vorletzten Schicht als Feature-Vektor. Dieser wurde normalisiert, mit einem lienaren SVM-Klassifizierer kombiniert und dann mit den

[11] Hierzu wurden die 1000 Klassen von ImageNet gesplittet (einmal zufällig und einmal thematisch), die eine Hälfte wurde für das Originalnetz genutzt und die andere Hälfte für das Zielnetz.

[12] Sofern die Datenmenge sehr groß ist, bedarf es logischerweise keines Transfer Learning, da man direkt auf dieser trainieren könnte.

[13] In diesem Fall wurden ConvNets mit 16-19 solcher Schichten untersucht; zum Vergleich: Das AlexNet umfasst 8 gewichtete Schichten.

Zieldaten trainiert. Auf diese Weise erzielten Simonyan und Zisserman einige State-of-the-Art-Resultate nicht nur in Bezug auf die Bildklassifizierung (selbe Zielaufgabe, andere Datensets), sondern auch bei der Action-Classification (andere Zielaufgabe) beim PASCAL-VOC-2012-Benchmark. Die Autoren machten ihr Modell der Öffentlichkeit zugänglich. Infolgedessen berichteten andere Mitglieder der Forschungsgemeinde über ihre Erfolge mit dem Modell, welches in Bezug auf ein großes Spektrum an Bilderkennungsaufgaben die bis dato etablierten, flacheren Architekturen übertraf. Zu diesen Aufgaben gehören u.a. Objektdetektion [Gi14], bei der State-of-the-Art-Ergebnisse erzielt werden konnten, semantische Segmentierung [LSD15] und Image Caption Generation [KSZ14, KFF15].

[Az15] stellt eine systematische Studie dar, die thematisiert, wie gelernte Repräsentationen bzw. Feature von Datensets wie ImageNet transferiert werden können. Dazu werden einerseits verschiedene Faktoren vorgestellt, andererseits werden differente Zielaufgaben untersucht. Die insgesamt 15 Zielaufgaben lassen sich 5 Hauptkategorien zuordnen, welche da lauten (mit zunehmender Distanz zur ursprünglichen Aufgabe der Bildklassifizierung mit ImageNet): Bildklassifizierung, Attributdetektion, Fine-grained Recognition, Compositional sowie Instance Retrieval. Die insgesamt 7 ermittelten Faktoren, die Einfluss auf die Transferierbarkeit haben, werden im Rahmen von Best Practices vorgestellt. Hierzu wird jeweils unter Betracht gezogen, wie weit die Zielaufgabe von der Quellaufgabe entfernt ist. In Bezug auf den ersten Faktor, Early Stoppage, verkünden die Autoren, dass man diese Methode wider Erwarten nicht durchführen sollte, da sie keinen messbaren Einfluss auf die Performance des ConvNets hatte. In Bezug auf den Faktor der Netztiefe empfehlen sie, diese so tief wie möglich zu gestalten. Bei der Netzbreite hingegen raten sie zu breiteren Schichten, sofern die Zielaufgabe nahe an der Quellaufgabe zu lokalisieren ist. Andernfalls solle die Breite moderat gewählt werden. Zu Fine-Tuning wird generell geraten, dabei ließen sich die höchsten Gewinne erzielen, wenn die Zielaufgabe weiter von der Quellaufgabe entfernt sei. Nichtsdestotrotz, so Azizpour et al., sei Fine-Tuning bei geschickter Wahl der Parameter immer wenigstens in kleinem Ausmaß hilfreich. Bezüglich der Reduktion der Dimensionalität empfehlen die Autoren, die ursprüngliche beizubehalten, sofern die Zielaufgabe nahe der Quellaufgabe ist. Ansonsten solle man die Dimensionlität reduzieren. Ein weiterer Faktor ist die Netzschicht, die für die Feature Extraction genutzt wird. Kohärent zu den Ergebnissen einiger der zuvor vorgestellten Arbeiten merken die Autoren an, dass die ersten Schichten generische Feature extrahieren und somit auch für Zielaufgaben geeignet seien, die weiter von der Quellaufgabe entfernt sind. Die tieferen Schichten hingegen extrahieren spezifische Feature, sodass diese entsprechend für quellnahe Aufgaben genutzt werden könnten. Als wichtigsten Faktoren führen Azizpour et al. die Quellaufgabe selbst an, da sich eine klare Korrelation der Performance bezüglich der verschiedene Aufgaben mit der Distanz zur Quellaufgabe zeige.

Die letzte Arbeit [SKP15], welche innerhalb dieses Abschnitts behandelt werden soll, hat für PROPOSE.AI eine besondere Relevanz. Vorgestellt wird das System FaceNet. Dieses ermöglicht eine Zuordnung von Gesichtsbildern zu einem euklidischen Raum, innerhalb dessen die Distanzen ein Maß für die Ähnlichkeit von Gesichtern darstellen. Auf dieser

Grundlage können dann verschiedene Aufgaben wie Gesichtsverifikation, Gesichtserkennung oder Clustering durchgeführt werden. Die Vorhergehensweise unterscheidet sich dabei grundlegend von den vorherigen Ansätzen für Gesichtserkennung. Diese (ein Beispiel hierfür ist das System „DeepFace“ [Ta14]) folgen prinzipiell dem Transfer-Learning-Ansatz, wie er in einigen der zuvor vorgestellten Arbeiten benutzt wurde: Ein ConvNet wird mit einer Menge von bekannten Gesichtsidentitäten trainiert, anschließend wird einer der (Zwischen-)Layer[14] als Repräsentation weiterverwendet. Diese soll eine Generalisierung der Erkennung ermöglichen, sodass auch neue Gesichter genutzt werden können, die nicht in der Trainingsmenge enthalten waren. Schroff et al. bezeichen diesen Zwischenlayer, aus dem die Feature extrahiert werden, als Bottleneck, da man hoffen müsse, dass die Repräsenation sich ausreichend gut generailsieren lasse, sodass die Adaption für neue Gesichter gegeben sei. Bei FaceNet hingegen wird der Output direkt als 128-dimensionale Einbettung[15] unter Verwendung einer Triplet-Loss-Funktion trainiert. Die FaceNet-Einbettungen können dann direkt als Feature-Vektoren genutzt werden, sodass die Bottleneck-Zwischenschicht vermieden wird. Nachfolgend soll die Triplet-Loss-Funktion vorgestellt werden: Grundsätzlich gesucht wird nach einer Einbettung $f(x)$ eines Bildes x in einen Feature Space $\mathbb{R}^d$, wobei d der Anzahl der Dimensionen des euklidischen Raums entspricht, sodass die quadrierte Distanz zwischen allen Gesichtern der selben Identität möglichst klein ist, während die quadrierte Distanz zwischen verschiedenen Gesichtern groß sein soll. Das Triplet besteht nun aus einem Ankerbild x_i^a, einem Positivbild x_i^p sowie einem Negativbild x_i^n. Das Ankerbild einer bestimmten Person soll nun näher zu allen anderen Bildern der selben Person sein als zu jedem der Bilder einer anderen Person. Mathematisch ausgedrückt stellt sich dies wie folgt dar:

$$\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2 \tag{1}$$

$$\forall \big(f(x_i^a), f(x_i^p), f(x_i^n)\big) \in T \tag{2}$$

α stellt hierbei einen Abstand dar, der zwischen Positiv- und Negativpaare erzwungen wird. Die Loss-Function, welche letztendlich zur Bestimmung der Gradienten genutzt wird, ergibt sich dann als

$$L = \sum_i^N \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+ \tag{3}$$

wobei N der Kardinalität der Menge T aller möglichen Triplets entspricht. Angemerkt sei noch, dass die Auswahl der Triplets wichtig ist, um eine schnelle Konvergenz bei der Minimierung der Funktion sicherzustellen. Triplets, die den Constraint aus Gleichung 1 leicht erfüllen, tragen nicht zum Training bei und resultieren damit in einer langsameren Konvergenz. Gesucht wird deswegen nach harten Triplets, die den Contraint aus 1 verletzen. Zum Finden solcher Triplets stellen Schriff et al. eine Methode vor, ihre genaue Erläuterung soll allerdings nicht mehr Gegenstand dieses Abschnitts werden, sondern kann bei Interesse

[14] Der letzte Layer wird zur aufgabenspezifischen Klassifizierung genutzt.

[15] Unter einer Einbettung (engl. embedding) versteht man eine Zuordnung von diskreten Objekten zu einem Vektor bestehend aus reellen Zahlen [Te18]. Ein diskretes Objekt könnte in diesem Fall beispielsweise ein Gesicht sein.

in der Arbeit nachgelesen werden. Mit ihrem Modell erzielen Schriff et al. State-of-the-Art-Ergebnisse beim Labeled-Faces-in-the-Wild-Datenset (LFW) mit einer Genauigkeit von 99,63% (und übertreffen damit die menschliche Performance von 97,53% [Le16]). Um Aufgaben wie Gesichtsverifikation anzugehen, können die FaceNet-Einbettungen als Feature-Vektoren genutzt werden.

4 Rückschlüsse für PROPOSE.AI

In diesem letzten Abschnitt sollen die verschiedenen Erkenntnisse Revue passiert und auf ihre Bedeutung für die Projektgruppe PROPOSE.AI bezogen werden. Zunächt lässt sich allgemein festhalten, dass ConvNets aktuell den State of the Art in Bezug auf Computer Vision darstellen. Bereits für die ersten beiden Anwendungsfälle, die umgesetzt werden sollen, namentlich Brillenpassreader und Brillenberater, bietet es sich an, diese Art der neuronalen Netze zu benutzen, da in beiden Fällen Fotos verarbeitet und Feature extrahiert werden sollen. In Bezug auf ein Gesicht beispielsweise könnten solche Feature von einzelnen Kanten und Konturen in den frühen Schichten über Gesichtsmerkmale wie Nase, Mund etc. in den mittleren Schichten bis hin zu vollständigen Gesichtern in den tiefen Schichten reichen. Die verschiedenen vorgestellten Paper legen die Empfehlung nahe, vortrainierte Netze zu nutzen und auf unsere Anwendungsfälle zuzuschneiden. Dies kann damit begründet werden, dass unsere Ressourcen (Zeit, GPU, Daten?) begrenzt sind, sodass wir es uns z.B. vermutlich nicht erlauben können, ein ConvNet mehrere Tage bis hin zu Wochen zu trainieren, wie dies bei einigen der vorgestellten Netze der Fall war. Eine weitere Erkenntnis ist, dass die Beschaffung von ausreichenden Datenmengen für Training, Validierung und Testen der Architekturen elementar ist, vor allem um die Gefahr des Overfitting zu vermeiden. Bei der Wahl des vortrainierten Netzes wird es wichtig sein, eines zu benuzten, welches nahe an der Zielaufgabe und den Zieldaten ist. So etwa eignet sich in Bezug auf den Brillenberater das FaceNet-Modell (dessen Implementierung z.B. in Tensorflow gefunden werden kann), da dieses bereits für Aufgaben mit Gesichtsbildern spezialisiert ist. Fine-Tuning sollten wir soweit mögich durchführen, die penible Feinjustierung einzelner Hyperparameter im Vorfeld hingegen scheint zunächst eher weniger relevant, da wir keine Forschungswettbewerbe gewinnen wollen. Überdies sollten wir uns intensiv mit der Triplet-Loss-Funktion beschäftigen und diese nutzen, da sie momentan den State of the Art darstellt und dazu beiträgt, dass ConvNets z.B. sogar die menschliche Performance bei der Gesichtserkennung (LFW-Datenset) übertreffen. Diese gesammelten Erkenntnisse sind dabei nicht nur auf unsere ersten beiden Anwendungsfälle beschränkt, sondern lassen sich analog auf andere potenzielle Services innerhalb unseres Service Layers übertragen, so zum Beispiel die Trendanalyse für Brillen unter Verwendung von Twitter-Fotos.

Literaturverzeichnis

[Az15] Azizpour, Hossein; Razavian, Ali Sharif; Sullivan, Josephine; Maki, Atsuto; Carlsson, Stefan: From generic to specific deep representations for visual recognition. In: CVPRW

DeepVision Workshop, June 11, 2015, Boston, MA, USA. IEEE conference proceedings, 2015.

[BSF94] Bengio, Yoshua; Simard, Patrice; Frasconi, Paolo: Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks, 5(2):157–166, 1994.

[De09] Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09. 2009.

[GBC16] Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron: Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org.

[Gi14] Girshick, Ross; Donahue, Jeff; Darrell, Trevor; Malik, Jitendra: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. S. 580–587, 2014.

[Ka18] Karpathy, Andrej: , CS231n: Convolutional neural networks for visual recognition. http://cs231n.github.io/, 2018.

[KFF15] Karpathy, Andrej; Fei-Fei, Li: Deep visual-semantic alignments for generating image descriptions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. S. 3128–3137, 2015.

[KSH12] Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. S. 1097–1105, 2012.

[KSZ14] Kiros, Ryan; Salakhutdinov, Ruslan; Zemel, Richard S: Unifying visual-semantic embeddings with multimodal neural language models. arXiv preprint arXiv:1411.2539, 2014.

[Le89] LeCun, Yann; Boser, Bernhard; Denker, John S; Henderson, Donnie; Howard, Richard E; Hubbard, Wayne; Jackel, Lawrence D: Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4):541–551, 1989.

[Le16] Learned-Miller, Erik; Huang, Gary B; RoyChowdhury, Aruni; Li, Haoxiang; Hua, Gang: Labeled faces in the wild: A survey. In: Advances in face detection and facial image analysis, S. 189–248. Springer, 2016.

[LSD15] Long, Jonathan; Shelhamer, Evan; Darrell, Trevor: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. S. 3431–3440, 2015.

[Ra14] Razavian, Ali Sharif; Azizpour, Hossein; Sullivan, Josephine; Carlsson, Stefan: CNN features off-the-shelf: an astounding baseline for recognition. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on. IEEE, S. 512–519, 2014.

[RW17] Rawat, Waseem; Wang, Zenghui: Deep convolutional neural networks for image classification: A comprehensive review. Neural computation, 29(9):2352–2449, 2017.

[SKP15] Schroff, Florian; Kalenichenko, Dmitry; Philbin, James: Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE conference on computer vision and pattern recognition. S. 815–823, 2015.

[Sp14] Springenberg, Jost Tobias; Dosovitskiy, Alexey; Brox, Thomas; Riedmiller, Martin: Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806, 2014.

[Sr14] Srivastava, Nitish; Hinton, Geoffrey; Krizhevsky, Alex; Sutskever, Ilya; Salakhutdinov, Ruslan: Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):1929–1958, 2014.

[SZ14] Simonyan, Karen; Zisserman, Andrew: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[Ta14] Taigman, Yaniv; Yang, Ming; Ranzato, Marc'Aurelio; Wolf, Lior: Deepface: Closing the gap to human-level performance in face verification. In: Proceedings of the IEEE conference on computer vision and pattern recognition. S. 1701–1708, 2014.

[Te18] TensorFlow: , Embeddings. `https://www.tensorflow.org/versions/master/programmers_guide/embedding`, 2018.

[Yo14] Yosinski, Jason; Clune, Jeff; Bengio, Yoshua; Lipson, Hod: How transferable are features in deep neural networks? In: Advances in neural information processing systems. S. 3320–3328, 2014.

[ZF14] Zeiler, Matthew D; Fergus, Rob: Visualizing and understanding convolutional networks. In: European conference on computer vision. Springer, S. 818–833, 2014.

Convolutional Neural Networks - Landmark Prediction and Region Proposal

Lea Lohmann[1]

Abstract: Diese Arbeit beschäftigt sich mit Convolutional Neural Networks und deren Anwendung in den Bereichen der Lokalisation von Landmarks und der Object Detection. Zunächst werden in einer Einführung die Funktionsweise und der Aufbau von ConvNets kurz beschrieben. Hierauf folgt eine genauere Betrachtung der Landmarks, wobei sich hauptsächlich auf die Orientierungspunkte im menschlichen Gesicht sowie deren Auffinden durch die Nutzung von ConvNets konzentriert wird. Hierbei werden zusätzlich Optimierungsmöglichkeiten dargestellt. Im Anschluss wird das Object Detection Verfahren vorgestellt und in seine Kategorien unterteilt. Für die Algorithmen basierend auf Region Proposal wird der Faster R-CNN Algorithmus mit dem Region Proposal Network erläutert. Als Stellvertreter der regressionbasierten Algorithmen wird auf YOLO eingegangen. Schlussendlich wird die Relevanz der vorgestellten Themen im Bezug auf die Projektgruppe PROPOSE.AI eingeschätzt.

Keywords: Convolutional Neural Network; Landmark Prediction; Facial Landmark Location; Region Proposal; Deep Learning; Region Proposal Network; Faster R-CNN; YOLO; Object Detection

1 Convolutional Neural Network

Ein neuronales Netz ist eine Ansammlung von einzelnen Neuronen, die Einheiten zur Informationsverarbeitung darstellen und schichtweise in einer Netzarchitektur angeordnet sind. Ein Convolutional Neural Network, auf deutsch etwa faltendes neuronales Netzwerk, kurz CNN oder ConvNet, ist das meist verbreiteste Deep Learning Neural Network. [Gr16, Chapter 5] Ein ConvNet ist ein Netzwerk, bestehend aus mehreren Layern, dass für die 2D Bildklassifizierung entworfen wurde.
Jeder einzelne Layer setzt sich aus mehreren 2D Ebenen, Planes, zusammen, die jeweils aus vielen unabhängigen Neuronen bestehen. Diese Neuronen besitzen lernbare Gewichte (weights) and neuronen-spezifische Verzerrung (bias). Die Gewichte beschreiben die Reichweite des Informationsflusses entlang einer Verbindung. Dazu werden sie jeweils von den Neuronen vergeben. Diese Gewichte werden nach der Addition der Verzerrung an die Neuronen der nächsten Schicht weiter gegeben.
Die Werte für Gewicht und Verzerrung werden zu Beginn des Trainings initialisiert. Für

[1] Universität Oldenburg, VLBA, Ammerländer Herrstraße 114, 26169 Oldenburg, Deutschland lea.lohmann@uni-oldenburg.de

das bestmögliche Ergebnis werden sie während des Trainingsprozesses an die Anforderungen angepasst. [Gr16, Chapter 5] Das traditionalle ConNet führt drei Operationen aus: Convolution, Sampling und Outputting. [Zh15]

1.1 Architektur

Ein ConvNet setzt sich im wesentlichen aus den folgenden Elementen zusammen:

Convolutional Layer Die Anzahl der Convolutional Layer in einem ConvNet ist abhängig von der Dimension und Komplexität des Problems. Der Input jedes Convolutional Layers wird durch die mathematische Funktion Konvolution verarbeitet. Die Konvolution kann als Produkt zweier Funktionen betrachtet werden und ist wie folgt definiert: $s(t) = (x*w)(t) = \sum_{n=-\infty}^{\infty} x(a)w(t-a)$, wobei das erste Argument x als Input und das zweite Argument w als Kernel oder auch Filter verstanden werden kann. Die Ausgabe wird als Feature Map bezeichnet, welche in tieferen Schichten wieder als Input für weitere Convolutional Layer fungiert. [GBC16, Chapter 9]
Bei einem Input der Dimension N x N x D, und des Kernels K x K x D, wobei D die Tiefe des eingegebenen Bildes ist, ist die Anzahl der Neuronen V jeder Reihe des Outputs gegeben durch V = (N - K + 2P) / S + 1. P stellt dabei die Anzahl der leeren Pixel dar, die im Zuge eines möglichen Paddings hinzugefügt werden können und S die Schrittgröße, stride, die die Entfernung zwischen benachbarten Filtern festlegt. [Gr16, Chapter 5]

Pooling Layer Pooling Layer reduzieren die Größe und Auflösung der Darstellung und des Lernaufwandes. Dadurch wird die Berechnungszeit beschleunigt und Overfitting kontrolliert. Pooling Layer arbeiten unabhängig auf jedem Inputteil. Innerhalb eines Pooling Layers findet kein Lernvorgang statt, da die ausgeführte Funktion fest implementiert ist.
Es gibt verschiedene Arten des Poolings. Die meistgenutzten sind Max- und Average Pooling. Beim Max Pooling wird das Maximum eines Bereiches zurück gegeben, während beim Average Pooling der Durchschnitt zurück gegeben wird. [Gr16, Chapter 5]

Fully Connected Layer Der Output eines ConvNets ist gegeben durch einen Fully Connected Layer, kurz FC Layer. Er liefert Lokalisations- und Klassifikationsentscheidungen. Die Gewichte der FC Layer werden während des Trainings bestimmt. Die Outputgröße entspricht 1 x 1 x n, wobei n die Nummer der zu klassifizierenden Klassen symbolisiert. [Gr16, Chapter 5]

Die einzelnen Layer setzten sich dreidimensional aus Breite (width), Höhe (height) und Tiefe (depth) zusammen. Die Tiefe wird bestimmt durch die Anzahl an parallelen Feature Maps. In Abb. 1 ist die Basisstruktur eines CNN zu sehen. [Gr16, Chapter 5]

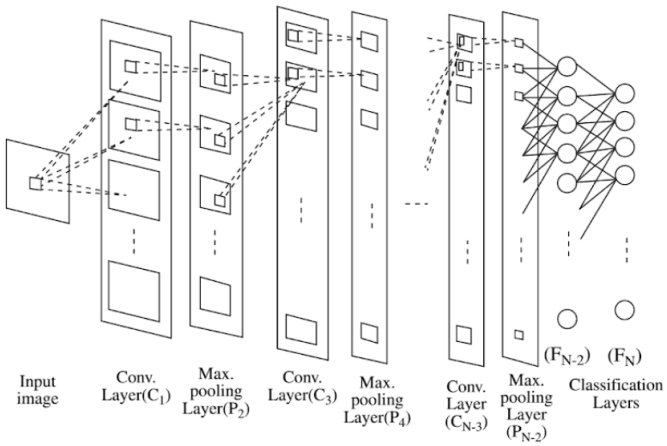


Abb. 1: Basisstruktur eines Convolutional Neural Network

2 Landmark Prediction

Als Landmarks, auf deutsch etwa Orientierungspunkte, werden wichtige Punkte in einem Bild verstanden. Diese können in vielen verschiedenen Bereichen der Bildverarbeitung zum Einsatz kommen. Eines der weitverbreitesten Gebiete, und im Zusammenhang mit unserer Projektgruppe besonders interessant, sind die Facial Landmarks, also Orientierungspunkte im Gesicht. Aus diesem Grund spezialisiere ich mich im Folgenden auf dieses Teilgebiete. Facial Landmarks besitzen eine semantische Bedeutung, wie zum Beispiel die Nasenspitze, die Mundwinkel oder die Mittelpunkte der Augen. Diese Punkte enthalten zusätzlich wichtige geometrische Informationen, die für andere Gesichtsanalyseverfahren, wie beispielsweise die Gesichtserkennung (Face Recognition), genutzt werden können.[Fe17]
Die Erkennung von Orientierungspunkten im Gesicht, Facial Landmark Detection, ist eine fundamentale Komponente in vielen Gesichtsanalysen. Das Ziel eines Facial Landmark Detection Verfahren ist das Auffinden der Koordinaten eines Sets von vorher definierten Orientierungspunkten in einem 2D Gesicht. [Fe17] Dieses Verfahren lässt sich in zwei Methodenkategorien einteilen: die namely regression based und die template fitting Methode. Während bei der namely regression based Methode die Lokalisation der Landmarks explizit durch eine Regression mit Bildfunktion geschätzt wird, werden in der template fitting Methode Gesichtsvorlagen erstellt, die als Maske auf die Eingabebilder gelegt werden, wodurch die Landmarks abgelesen werden können. [Zh14] Da die meisten Erkennungsverfahren im Deep Learning Bereich auf Regression basieren, beschäftigen wir uns in der Folge mit der namely regression based Methode.

2.1 Facial Landmark Lokalisation in ConvNet

Die Aufgabe eines ConvNet in der Erkennung von Facial Landmarks ist das Finden eines nichtlinearen Mappings $\Phi : I \rightarrow s$, wobei der shape Vektor $s \in \mathbb{R}^{2L} = (x_1, .., x_L, y_1, .., y_L)^T$ mit L der vorgegebenen Anzahl an Landmarks, die gefunden werden sollen und (x_L, y_L) die Koordinaten des L-ten Orientierungspunktes sind.
Eingegeben wird ein Bild $I \in \mathbb{R}^{HxWx3}$, welches häufig noch im Zuge der Bounding Boxes zugeschnitten wird. Die Tiefe erklärt sich durch die Anzahl der Kanäle (Channels) für farbige Bilder, da es einen roten, grünen und blauen Kanal gibt.
Sei zusätzlich $\Omega = \{l_i, si_i\}_{i=1}^N$ eine gelabelte Trainingsmenge. Das Ziel eines ConvNet ist es dann, ein Φ zu finden, dass die folgende Funktion minimiert: $\sum_{i=1}^N loss(\Phi(l_i), s_i)$ [Fe17]

2.2 Loss Function

Die Loss Funktion ist eine vordefinierte Funktion, die den Unterschied zwischen einem vorhergesagtem shape Vektor und seinem tatsächlichem Wert, seiner ground truth, berechnet. Die korrekte Gestaltung dieser Funktion ist entscheidend für das Gelingen des Facial Landmark Detection Verfahrens. Loss ist dabei wie folgt definiert, wobei s der ground truth shape Vektor der Facial Landmarks ist: $loss(s, s') = \sum_{i=1}^2 Lf(s_i, s_i')$. Grade bei der Erkennung von Orientierungspunkten im Gesicht muss aufgrund der in 2.3 genannten Probleme mit Lokalisationsfehlern gerechnet werden. Hier ist es die Aufgabe der Loss Funktion, diese Fehler schnell und bestmöglich auszubessern.
Die meisten Verfahren zur Erkennung von Facial Landmarks nutzen die L2 Loss Funktion, $L2(x) = (1/2) \cdot x^2$, aufgrund ihrer hohen Empfindlichkeit im Bezug auf Ausreißer. [Fe17]

2.3 Optimierungsmöglichkeiten

Facial Landmark Detection gilt nach wie vor als große Herausforderung. Durch Verdeckung von Gesichtsteilen oder die vielen möglichen Positionen eines Gesichtes wird diese Arbeit stark erschwert. Zusätzlich lassen sich die Ergebnisse durch Faktoren, wie eine ungenaue Ausgabe der Bounding Boxes eines Gesichtserkennungsverfahren oder in-plane head rotations, verschlechtern. [Fe17]
Eine Optimierung des Ergebnis lässt sich durch das Beachten und Miteinbeziehen von unterschiedlichen Faktoren, wie der Kopfhaltung, des Alters und Geschlechtes und die Bestimmung des Gesichtsausdruck, erreichen. Je mehr Faktoren mit einbezogen werden, umso besser verspricht der Output zu werden. So lachen zum Beispiel Kinder immer mit einem weit geöffnetem Mund, was bei der Bestimmung der Mundwinkelposition hilfreich ist. [Zh14]
In diesem Zusammenhang ist die Loss Funktion hilfreich. Die unterschiedlichen Verfahren haben meist keinen einheitlichen Output und setzen ihre Grenzen im Entscheidungsprozess

unterschiedlich. Des Weiteren unterscheiden sich die Aufgaben der Erkennungen in ihrem Anspruch, was zur Folge hat, dass einige Verfahren schneller zum Overfitting führen als andere. All diese Auswirkungen können durch eine aufgabenspezifische Loss Funktion behoben werden, die Ausschläge eindämmt und die Ergebnisse zusammenführt. [Zh14]
Außerdem führt die sogenannte iPose-based Data Balancing (PDB) Strategie zu einer Verbesserung der Genauigkeit. Diese ergründet die Schwierigkeiten der Erkennung bereits in den Trainingsdaten: Netzte werden häufig überwiegend mit Bildern vortrainiert, die Gesichter in der Frontalansicht zeigen. Daraus resultiert ein Datenungleichgewicht. Das Netz hat nach der Trainingsphase Schwierigkeiten, Bilder von Gesichtern in einer anderen Pose korrekt zu erkennen und die Orientierungspunkte zu bestimmen. Es ist folglich wichtig, bereits in den Trainingsdaten für ein ausgeglichenes Verhältnis zwischen den unterschiedlichen Gesichtshaltungen und Posen zu sorgen, damit das Netz bestmöglich vortrainiert werden kann. Beispielsweise können Trainingsdaten von Gesichtern in Mehrfachansicht gewählt oder auf die Nutzung eines 3D Gesichtmodells zurückgegriffen werden. [Fe17]

3 Object Detection

Menschen sind in der Lage, ein Bild zu betrachten und im selben Moment die im Bild abgebildeten Objekte und deren Standorte zu erkennen. Das visuelle System des Menschen arbeitet sehr genau und in Echtzeit. Dieses Ziel wird ebenfalls von Objekterkennungsverfahren (Object Detection) verfolgt.
Ein Erkennungsverfahren setzt sich dabei aus der Lokalisation und Klassifikation von mehreren Objekten zusammen. Die Klassifikation erkennt in einem Bild mit einem Objekt um was für ein Objekt es sich handelt. Die Lokalisation gibt eine örtliche Beschreibung eines Bildes zurück. Sei beispielsweise ein Bild mit einem Auto gegeben, so würde die Klassifikation das Objekt als Auto klassifizieren und die Lokalisation beschreiben, wo im Bild sich das Auto befindet.

Bei der Objekterkennung werden zur örtlichen Beschreibung eines Objektes nicht mehr nur einzelne Koordinaten angegeben, wie beim Landmark Detection Verfahren, sondern kleine Bereiche, in denen sich das Objekt befindet. Häufig werden für diese Lokalisation Begrenzungsboxen, sogenannte Bounding Boxes, verwendet. Diese sind rechteckig und werden durch die Koordinaten ihres Mittelpunktes sowie Höhe und Breite definiert. Ein mögliches Ergebnis eines Object Detection Verfahrens mit Bounding Boxes ist in Abbildung 2 zu sehen. Die einzelnen Objekte wurden klassifiziert (Schaf, Zeichen und Boot) und durch Begrenzungsboxen (türkis, blau und rot) lokalisiert.

Abb. 2: Beispielhaftes Ergebnis eines Objekterkennungsverfahren [BRC16]

Objekterkennungsverfahren mit ConvNets können in zwei Kategorien unterteilt werden [YZY17]: Die Algorithmen basierend auf Region Proposal, wie R-CNN, Fast R-CNN und Faster R-CNN, und die regressionsbasierten Algorithmen wie YOLO [Di15] und SSD [Li15]. Im Folgenden werden der Faster R-CNN und der YOLO Algorithmus näher vorgestellt.

3.1 Faster R-CNN

Die aktuellen Objekterkennungsalgorithmen mit der besten Performance basieren auf der Technik des Region Proposals zur Lokalisation der Objekte. Diese werden Region-based Convolutional Neural Networks genannt. Diese Verfahren ermitteln zunächst Bildbereichsvorschläge (Region Proposal), in denen dann Objekte klassifiziert und lokalisiert werden. [Re15]
Häufig enthält ein Bild viele Bereiche, in denen sich keine Objekte befinden. Durch Region Proposal werden die relevanten Bereiche eines Bildes, die semantisch bedeutsam sind und in Beziehung zu Objekten stehen, identifiziert. In diesen Teilen wird dann Object Detection durchgeführt. [Gi13]

Der repräsentativste Region-based ConvNet-Algorithmus ist der Faster R-CNN, welcher eine Weiterentwicklung von R-CNN und Fast R-CNN darstellt. Der Faster R-CNN Algorithmus löst das Flaschenhalsproblem des Region Proposals durch die Nutzung eines neuronalen Netzes. Ein Region Proposal Network (RPN, siehe 3.2) ermittelt Regionen, in denen sich Objekte befinden. In diesen Bereichen werden dann mittels Feature Extraction die Merkmale dieser Objekte berechnet, auf deren Basis anschließend die Klassifikation durchgeführt wird. Ausgegeben wird für jedes Objekt eine Klassenzugehörigkeit und eine Begrenzungsbox, in der sich das Objekt befindet. [YZY17]
Die einzige Problematik entsteht durch die eventuelle Ungenauigkeit der vorgeschlagenen Regionen eines RPN, wenn mit Anchor Boxes (siehe 3.2.2) gearbeitet wird, welche durch vorgegebene Skalierungen definiert sind. Durch diese starren Formen wird wenig Flexibilität geboten. [YZY17]

3.2 Region Proposal Network

Neben dem Selective Search Algorithmus[Ui13] stellt das Region Proposal Network, kurz RPN, eine Möglichkeit zur Ermittlung von Bildbereichsvorschlägen (Region Proposal) dar. Ein ConvNet extrahiert die Feature Map. Diese Convolutional Feature Map wird dann mit Hilfe von Sliding Windows (siehe 3.2.1) und, da in den einzelnen Fenstern mehrere Objekte erwartet werden, Anchor Boxes (siehe 3.2.2) verarbeitet. In Abbildung 3 ist der Aufbau eines RPNs zu sehen. Für k die maximale Anzahl an Objekten werden 2k Klassifizierungen, Objekt oder kein Objekt, und 4k Koordinaten der Boxen zurück gegeben. Positiv als Objekt klassifiziert werden Anchor Boxes mit dem höchsten IoU (siehe 3.2.3), sowie alle Boxen mit IoU $\geq$ 0,7. [Re15]

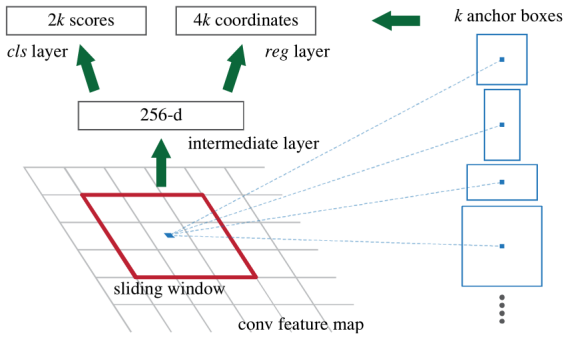


Abb. 3: Region Proposal Network [Re15]

Ausgegeben werden Bounding Boxes mit einem Objectness-Score. Der Score gibt die Wahrscheinlichkeit für die Existenz eines Objektes in der Box an.
Das RPN lernt während des Trainings, Vordergrund und Hintergrund zu unterscheiden und optimale Begrenzungen um Objekte zu setzten. Dass sich das RPN und das CNN für die Objektklassifizierung viele Layer teilen können und damit mehrfache Berechnungen eingespart werden können, gilt als großer Vorteil der RPNs. Das Training des Region Proposal Networks kann parallel zu dem des ConvNets stattfinden, wodurch keine zusätzliche Trainingszeit anfällt. [Re15]

3.2.1 Sliding Window

Die Sliding Window Methode dient der Verbesserung der Lokalisation von Objekten. Dafür wird ein rechteckiges Fenster einer festen Größe über das eingegebene Bild verschoben. Diese Verschiebung geschieht in einer festgelegten Schrittgröße (stride) s, die ebenfalls zu Beginn definiert wird. Je kleiner s ist, desto präziser wird das Ergebnis. In jedem der dadurch entstehenden Fenster wird Objekt Detection durchgeführt. Das Ziel ist das Auffinden eines

minimalen Fensters, einer Bounding Box, in der sich ein Objekt befindet. Der große Nachteil dieses Verfahrens ist die Höhe der anfallenden Berechnungskosten. Eine große Schrittweite könnte diese Kosten senken, allerdings würde in diesem Fall auch die Performance leiden. Die Lösung ist die konvolutionale Implementierung der Sliding Windows, wodurch doppelte Berechnungen eingespart werden. Dabei können alle Berechnungen, die in mehreren Fenstern benötigt werden, einmalig durchgeführt und deren Ergebnisse in allen Fenstern genutzt werden.[Se13]

3.2.2 Anchor Boxes

Befinden sich in einem Fenster, das auf Objekte untersucht werden soll, mehrere Objekte, sind Anchor Boxes hilfreich. Anchor Boxes sind vordefinierte Rechtecke mit einem festen Mittelpunkt, wie beispielsweise in Abbildung 3 gezeigt. Für ein gefundenes Objekt wird überprüft, welche Anchor Box das Objekt am besten abdeckt. Objekte werden dann zusätzlich zu dem Bereich, in dem sie sich befinden, der Anchor Box mit dem höchsten IoU zugeordnet.
Die Anchor Box Methoden kann den Algorithmus zusätzlich unterstützt eine bessere Spezialisierung zu erlernen. So kann der Algorithmus beispielsweise lernen, dass stehende Anchor Boxes häufig Menschen und liegende Boxen häufig Autos enthalten. [Re15]

3.2.3 Intersection over Union

Zur Evaluation der Lokalisation dient der IoU-Wert (Intersection over Union). Dieser misst die Überlappung von zwei Begrenzungsboxen. Um die Genauigkeit zu ermitteln, wird die zurückgegebene Box mit der tatsächlichen Box, der ground truth, verglichen. Die IoU wird durch die Division der Schnittmenge durch die Vereinigung der beiden Boxen berechnet. Der resultierende Wert liegt zwischen 0 und 1. Je größer der IoU, desto präziser ist die zurückgegebene Box gewählt. Laut aktueller Konvention gilt eine Bounding Box für einen IoU-Wert $\geq$ 0,5 als richtig gesetzt. [Re15]

3.3 YOLO Algorithmus

Das Ziel des YOLO-Algorithmus (you only look once) ist die Optimierung der Genauigkeit von Bounding Boxes, da häufig die Größe und die Form nicht optimal sind. Bei dem YOLO-Algorithmus handelt es sich um einen der effektivsten Object Detection Algorithmen.
Für die Durchführung wird ein Bild mittels eines Gitters (Grid) in mehrere kleine Zellen (Gridcells) zerlegt. In jeder Zelle wird dann durch ein ConvNet gleichzeitig ein Lokalisations- und Klassifikationsalgorithmus durchgeführt, der einen Vektor zurück gibt. Dieser enthält die Klassifizierung sowie die fünf zur Bounding Box gehörenden Parameter x, y, w, h und den confidence score. Dieser gibt an, wie sicher es ist, in der angegebenen Box ein Objekt

gefunden und die Maße der Box korrekt gewählt zu haben.
Ein Objekt wird der Zelle zugeordnet, in der sich sein Mittelpunkt befindet. Die Koordinaten der Bounding Boxes werden in Relation zu der Gitterzelle angegeben. In jeder Zelle wird die obere linke Ecke als (0,0) und die untere rechte Ecke als (1,1) deklariert. Die Koordinaten des Mittelpunktes der Bounding Box (x,y) liegen demnach in dem Intervall [0;1]. Da die Bounding Box über die Größe der Gitterzelle hinaus gehen kann, sind die Werte der Höhe h und Breite w nicht beschränkt.
Das Gitter wird möglichst feingranular gewählt, gängig ist ein 19x19 Gitter. Die Effizienz wird durch die konvolutionale Implementierung und die dadurch geteilten Berechnungen gesichert.
Der YOLO-Algorithmus ist, im Gegensatz zum Faster R-CNN Algorithmus, in der Lage, in Echtzeit Objekte zu erkennen, wie beispielsweise in einem Video. YOLO ist also extrem schnell. Dafür hängt YOLO im Bezug auf die Genauigkeit anderen state-of-the-art Erkennungssystemen hinterher, da es grade bei kleinen Objekten Probleme hat diese präzise zu lokalisieren. [Di15]

3.3.1 Non-max Suppression

Ein mögliches Problem, dass bei der Objekterkennung durch YOLO auftreten kann, ist die Mehrfacherkennung von Objekten. Durch die Wahl eines feingranularen Gitters ist es möglich, dass ein Objekt in mehreren Zellen vorkommt und jede Zelle annimmt, den Mittelpunkt des Objektes zu enthalten. Es besteht also die Gefahr, dass in mehreren Zellen das selbe Objekt erkannt wird. Diese Mehrfacherkennung kann durch non-max suppression behoben werden.
Dafür wird die Begrenzungsbox mit dem höchsten confidence score für das Objekt betrachtet. Alle weiteren Bounding Boxes, die diese überlappen, werden eliminiert. Dieser Vorgang wird für alle Objekte wiederholt. Übrig bleibt für jedes Objekt genau eine Bounding Box. [Di15]

4 Praxisbezug PROPOSE.AI

Im Bezug auf die Projektgruppe PROPOSE.AI generiert die Nutzung von ConvNets, und hier speziell die Möglichkeit der Erkennung von Facial Landmarks, einen großen Mehrwert. Das aktuelle Ziel der Projektgruppe ist es einen Brillenberater sowie einen Brillenpass Reader zu entwickeln. Der Brillenberater soll der optimalen Beratung beim Online Brillenkauf dienen. Dazu besteht die Möglichkeit für den Kunden, ein Foto seines Gesichtes hochzuladen und anhand von Gesichtsform, Hautfarbe, Entfernung der Augen und vielen weiteren Merkmalen passende Brillen vorgeschlagen zu bekommen. Um diese Typberatung durchzuführen, ist eine Extraktion der wichtigsten Gesichtsmerkmale unumgänglich. Dazu kann das Facial Feature Detection Verfahren mittels ConvNets genutzt werden.
Ein mögliches Einsatzgebiet des Objeckt Detection Verfahrens mittels Faster R-CNN ist

das Auslesen eines Brillenpasses. Hierfür kann ein Kunde ein Foto seines Brillenpasses hochladen um seine Brillenwerte automatisch eintragen zu lassen. Da sich die einzelnen Werte in verschiedenen Brillenpässen immer an ähnlichen Stellen befinden, kann man ein Netz dahingehend trainieren und die Werte somit einfacher und schneller lokalisieren.
Ein weiteres Ziel für die Zukunft ist die Anprobe von Brillen über das Smartphone in Echtzeit. Das heißt, dass der Kunden über die Anwendung seine Kamera öffnet und Brillen aufgesetzt bekommt. Hierfür sind erneut die Landmarks von hoher Bedeutung, damit die jeweilige Brille den richtigen Platz in jedem Gesicht findet. Zusätzlich kann der YOLO Algorithmus durch seine Objekterkennung in Echtzeit zur Umsetzung beitragen.

Literaturverzeichnis

[BRC16] Bappy, Jawadul H.; Roy-Chowdhury, Amit K.: CNN based region proposals for efficient object detection. Image Processing (ICIP), 2016.

[Di15] Divvala, Santosh; Redmon, Joseph; Girshick, Ross; Farhadi, Ali: You Only Look Once: Unified, Real-Time Object Detection. CoRR, 15.

[Fe17] Feng, Zhen-Hua; Kittler, Josef; Awais, Muhammad; Huber, Patrik; Wu, Xiao-Jun: Wing Loss for Robust Facial Landmark Localistion with CNNs. CoRR, 17.

[GBC16] Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron: Deep Learning. MIT Press, `http://www.deeplearningbook.org`, 2016.

[Gi13] Girshick, Ross B.; Donahue, Jeff; Darrell, Trevor; Malik, Jitendra: Rich feature hierarchies for accurate object detection and segmentation. CoRR, 2013.

[Gr16] Graupe, Daniel: Kapitel 5. World Scientific Publishing Co. Pte. Ltd., 2016.

[Li15] Liu, Wei; Anguelov, Dragomir; Erhan, Dumitru; Szegedy, Christian; Reed, Scott; Fu, Cheng-Yang; Berg1, Alexander C.: SSD: Single Shot MultiBox Detector. CoRR, 2015.

[Re15] Ren, Shaoqing; He, Kaiming; Girshick, Ross; Sun, Jian: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. CoRR, 2015.

[Se13] Sermanet, Pierre; Eigen, David; Zhang, Xiang; Mathieu, Michael; Fergus, Rob; LeCun, Yann: OverFeat: Integrated Recognition, Localzation and Detection using Convolutional Networks. CoRR, 2013.

[Ui13] Uijlings, J.R.R.; van de Sande, K.E.A.; Gevers, T.; Smeulders, A.W.M.: Selective Search for Object Recognition. International Journal of Computer Vision, 2013.

[YZY17] Yuan, Peng; Zhong, Yangxin; Yuan, Yang: Faster R-CNN with Region Proposal. CS231n: Convolutional Neural Networks for Visual Recognition, 2017.

[Zh14] Zhang, Zhanpeng; Luo, Ping; Loy, Chen Change; Tang, Xiaoou: Facial Landmark Detection by Deep Multi-task learning. In: Computer Vision - ECCV 2014. Springer International Publishing, S. 94–108, 2014.

[Zh15] Zhang, Yuanyuan; Zhao, Dong; Sun, Jiande; Zou, Guofeng; Li, Wentao: Adaptiv CNN and ist applikation in Face Recognition. Neural Processing Letters, 2015.

Facial Feature Extraction

Wiebke Meyer[1]

Abstract: In dieser Arbeit wird das Thema "Facial Feature Extraction" behandelt. Übersetzt ins Deutsche bedeutet der Begriff "Extraktion von Gesichtsmerkmalen". Dabei handelt es sich um den Prozess zum Extrahieren von Gesichtsmerkmalen wie z.B. Augen, Augenbrauen, Mund, Nase und Kinn. Die Gesichtsmerkmalsextraktion stellt bei der Entdeckung von Gesichtern, der Ausdruckserkennung und der Gesichtserkennung einen der wichtigsten Prozesse dar. Um die Merkmale zu extrahieren, können verschiedene Methoden eingesetzt werden. Hierbei gibt es eine Einteilung der Ansätze in vier grundlegende Kategorien: "geometry-based", "template-based", "apperance-based" und "color-based". Neben diesen Ansätzen wird in dieser Arbeit auch die Extraktion von Gesichtsmerkmalen durch neuronale Netze vorgestellt.

Keywords: Facial Feature Extraction; Facial Features; Gesichtsmerkmale; Gesichtsmerkmalsextraktion; Face Detection; Face Recognition; Gesichtsentdeckung; Gesichtserkennung

1 Einleitung

In der heutigen Zeit gewinnt die Gesichtserkennung in unterschiedlichen Bereichen immer mehr an Bedeutung. Wo vor Jahren das Entsperren seines Smartphones nur mit der Eingabe eines Codes oder eines Musters möglich war, reicht heute schon teilweise ein Blick auf den Bildschirm, um das Smartphone durch Erkennung des Gesichts zu entsperren. Auch in Anwendungen wie z.B. Facebook, Snapchat oder Instagram sind Funktionen, für die das Gesicht von großer Bedeutung ist, vorhanden. Bei der Markierung von Bildern in Facebook zum Beispiel, werden die erkannten Gesichter in einem Bild vorgeschlagen, um sie mit dem Namen einer Person zu versehen. Bei Instagram und Snapchat gibt es die sogenannten Filter, mit denen das Gesicht in einem Bild oder Video "geschmückt" werden kann. Hierbei werden einem bspw. lange Wimpern, Sommersprossen oder eine Brille auf das Gesicht gezaubert. Damit solche Filter angewendet werden können und richtig positioniert werden, muss das Gesicht mit den einzelnen Merkmalen erkannt werden. Wie Informationen über die Gesichtsmerkmale gewonnen werden können, wird in dieser Arbeit behandelt.

Der Aufbau ist wie folgt strukturiert: In Abschnitt 2 wird eine grundlegende Einführung in das Thema gegeben, worauf in Abschnitt 3 die Vorstellung verschiedener Methoden für die Extraktion von Gesichtsmerkmalen folgt. Hier wird neben den vier Kategorien der

[1] Carl von Ossietzky Universität Oldenburg, VLBA, Ammerländer Heerstraße 114, 26169 Oldenburg, Deutschland
wiebke.meyer1@uni-oldenburg.de

Grundansätze auch ein weiteres Beispiel einer Methode zur Gesichtsmerkmalsextraktion gegeben. Im vierten Abschnitt werden ein Web-Service für die Gesichtsentdeckung und -erkennung, und ein Anwendungsbeispiel, in denen die Extraktion von Gesichtsmerkmalen Bestandteil ist, vorgestellt. Der letzte Abschnitt stellt einen Bezug zu der Projektgruppe PROPOSE.AI her, in dessen Rahmen diese Arbeit geschrieben wurde.

2 Grundlagen

Facial Feature Extraction, ins Deutsche als die Extraktion von Gesichtsmerkmalen übersetzt, ist der Prozess zum Extrahieren der Eigenschaften von Gesichtskomponenten aus einem Bild des menschlichen Gesichts [BK16]. Es handelt sich hierbei also um nichts anderes als das Identifizieren der genauen Lage verschiedener Gesichtsmerkmale. Dies umfasst unter anderem die Erkennung von Augen, Augenbrauen, Mund, Nase und Kinn [Wu12]. Die Erkennung und Lokalisierung der Augen ist unter allen Gesichtsmerkmalen von wesentlicher Bedeutung. Ausgehend hiervon werden alle anderen Gesichtsmerkmale identifiziert [BK16]. Die Gesichtsmerkmale können von unterschiedlichem Typ sein. Die Unterteilung der Typen erfolgt dabei in Region, Kernpunkt (Landmark) und Kontur/Umriss. Die Kernpunkt-Merkmale liefern im Gegensatz zu den regionsbasierten Merkmalen für Abgleichzwecke eine genauere und konsistentere Darstellung [BR08].

Ein Mensch ist in der Lage, in seinem ganzen Leben fast tausend Gesichter zu erkennen. Auch das Unterscheiden von Gesichtern ist ihm ohne große Schwierigkeiten möglich [Wu12]. Bei der Wahrnehmung des Gesichts spielen die menschlichen Gesichtsmerkmale eine bedeutende Rolle, und für die Erkennung gehören laut neurophysiologischer Forschung und Studien, Augen, Mund und Nase zu den wichtigsten Merkmalen. Wird ein menschliches Gesicht als ein Bild dargestellt, so ist es für diese Merkmale typisch, dass sie charakteristische Eigenschaften darstellen, was bei anderen Gesichtskomponenten wie Stirn, Wangen und Kinn nicht der Fall ist [BR08].

Die Gesichtsmerkmalsextraktion ist einer der wichtigsten Prozesse bei der Gesichtserkennung (engl. “face recognition“), der Ausdruckserkennung (engl. “expression recognition“) sowie bei der Gesichtsentdeckung (engl. “face detection“) [Hu09]. Bei der Gesichtserkennung handelt es sich um eine biometrische Methode, durch die Personen anhand von Merkmalen des Gesichtes identifiziert werden. Das Gesichtserkennungssystem sollte fähig sein, mit Hilfe einer vorab gespeicherten Bilddatenbank, eine oder sogar mehrere Personen in einem Bild zu identifizieren oder zu verifizieren [BR08].

Vor der Durchführung einer Gesichtserkennung, sollte von dem System ermittelt werden, ob ein Gesicht im gegebenen Bild oder Video vorhanden ist oder nicht. Bei diesem Vorgang handelt es sich um die Gesichtsentdeckung [BR08]. Diese beinhaltet die Zweiteilung des Bildes in einen Teil, der das entdeckte Gesicht enthält, und einen anderen Teil für den Hintergrund, wodurch der Gesichtsbereich für die Gesichtserkennung isoliert wird [BS14, BR08]. Der Vorgang der Gesichtsentdeckung wird oft zu gleicher Zeit mit der

Extraktion von Merkmalen durchgeführt [BR08]. Zum Erkennen einer Person benötigt es somit drei hauptsächliche Prozesse [Hu09]. Die Abbildung 1 zeigt den Gesamtprozess.

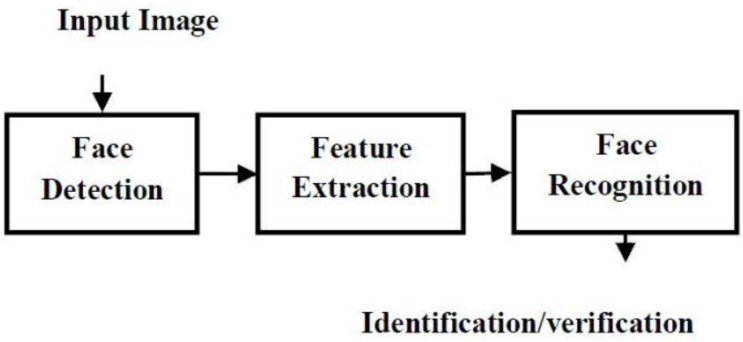


Abb. 1: Prozess der Gesichtserkennung [BS14]

Für die drei in Abbildung 1 abgebildeten Phasen des Gesamtprozesses werden mehrere unterschiedliche Methoden benötigt [BS14]. Die der Gesichtsentdeckung und -erkennung sollen im Folgenden kurz vorgestellt werden. Für die Methoden der Merkmalsextraktion erfolgt in einem eigenen Kapitel, siehe Abschnitt 3, eine detailliertere Beschreibung.

Methoden der Gesichtsentdeckung

Die Techniken der Gesichtsentdeckung können in eine merkmalsbasierte und bildbasierte Kategorie unterteilt werden. Merkmalsbasierte Ansätze verwenden für den Entdeckungsprozess die Gesichtsmerkmale [BS14]. Dies ist ein sehr trivialer Ansatz, da es durch die Unvorhersehbarkeit von Gesichtern und Umgebungsverhältnissen zu Beeinträchtigungen kommen kann. Bildbasierte Methoden sind robustere Techniken, die in unfreundlichen Umgebungen, wie z.B. der Entdeckung mehrerer Gesichter bei störungsintensiven Hintergründen, eingesetzt werden können [BS14].

Methoden der Gesichtserkennung

Die verschiedenen Ansätze der Gesichtserkennung können in drei Hauptgruppen eingeteilt werden. Hier gibt es ganzheitliche, merkmalsbasierte und hybride Ansätze. Bei den ganzheitlichen Ansätzen werden die Informationen des gesamten Gesichtsbereichs als Eingangsdaten für das System berücksichtigt. Anders ist dies in den merkmalsbasierten Methoden, bei denen die lokalen Merkmale wie Nase und Augen, und ihre geometrischen Beziehungen analysiert werden [BS14]. Hybride Ansätze verwenden die Kombination von lokalen und ganzen Merkmalen, orientiert an dem menschlichen Sehsystem, welches sowohl lokale Merkmale als auch das gesamte Gesicht wahrnimmt [BS14, BR08].

3 Methoden zur Gesichtsmerkmalsextraktion

Bei der Gesichtsmerkmalsextraktion werden Merkmalspunkte wie Augen, Nase und Mund extrahiert und dann als Eingabedaten verwendet. Um diese Gesichtspunkte bzw. Merkmale aus Bildern oder Videosequenzen von Gesichtern zu extrahieren, gibt es verschiedene Ansätze [BRU09, BS14]. In der Literatur reichen die beschriebenen Ansätze von der geometrischen Beschreibung auffälliger Gesichtsmerkmale bis hin zur Erweiterung digitaler Gesichtsbilder auf einer geeigneten Bildgrundlage [Hu09].

3.1 Basisansätze

Die unterschiedlichen Ansätze aus der Literatur können in die vier Kategorien “geometry-based“, “template-based“, “appearance-based“ und “color-based“ unterteilt werden [Hu09]. Nachfolgend werden diese grundlegenden Ansätze im Einzelnen beschrieben.

3.1.1 geometry-based

In diesen Ansätzen werden die Merkmale extrahiert, indem geometrische Informationen verwendet werden. Zu diesen Informationen gehören z.B. die relativen Positionen und Größen der Gesichtskomponenten [Hu09]. Somit zeigen die geometrischen Gesichtsmerkmale die Form und Lage der Gesichtskomponenten wie bspw. Mund, Augen, Augenbrauen und Nase. Um einen Merkmalsvektor, der die Gesichtsgeometrie darstellt, zu bilden, werden die Gesichtskomponenten oder -merkmalspunkte extrahiert [DS14]. Die Abbildung 2 zeigt die geometrische Darstellung einer Person.

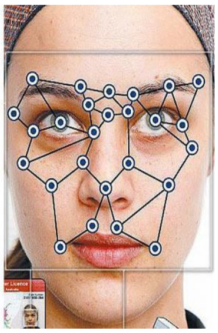


Abb. 2: Geometrische Darstellung einer Person [BS14]

Nach [BS14] können in dem geometriebasierten Ansatz zwei Methoden unterschieden werden. Nach der ersten Methode dieses Verfahrens, werden zunächst Ausrichtung und Ränder der wichtigsten Komponente erfasst, und ausgehend hiervon Merkmalsvektoren gebildet [BS14]. Die Merkmale werden von wesentlichen vertikalen und horizontalen

Projektionen aus dem Originalbild extrahiert. Dabei wird die horizontale “edge map“ verwendet, um sowohl die linke und rechte Gesichtsgrenze als auch die Nase zu extrahieren. Um Augen, Mund und Nasenwurzel zu extrahieren, wird die vertikale “edge map“ angewendet [YN02].

Als zweites gibt es in dieser Gruppe Methoden, die auf der Graustufen-Differenz unwichtiger und wichtiger Komponenten basieren, indem sogenannte Feature-Blöcke verwendet werden um die Graustufenverteilung in den Merkmalen zu ändern. Ein Gesichtsbild wird dabei in Blöcke unterteilt, wovon jeder sein entsprechendes Hauptpixel hat. Auf Basis des Graustufenwertes des Hauptpixel werden dann die Nachbarpixel untersucht und anschließend auf 0 oder 1 gesetzt. Für jede Region werden danach Histogramme erstellt, welche dann zu einem Merkmalsvektor für das Gesichtsbild kombiniert werden [BS14].

3.1.2 template-based

In vorlagebasierten Ansätzen werden die Gesichtsmerkmale unter Verwendung einzelner oder mehrerer Vorlagen, oder zum Teil sogar mit verformbaren, flexiblen Schablonen erkannt und beschrieben, und unter Verwendung einer geeigneten Energiefunktion extrahiert [BS14, YN02]. Hierbei wird das Merkmal, welches von Interesse ist, wie bswp in Abbildung 3 ein Auge, durch eine mit einem Parameter versehende Vorlage beschrieben. Parametrisierte Vorlagen ermöglichen ein grundsätzliches Wissen über die erwartete Form der Merkmale und können so den Erkennungsprozess führen. Bei der Durchführung des Vorlagenabgleiches mit einem Bild, verformt sich die Vorlage selbst, um die beste Anpassung zu finden [BS14]. Ihre Größe und Form können die Vorlagen so ändern, dass sie passend übereinstimmen [YN02].

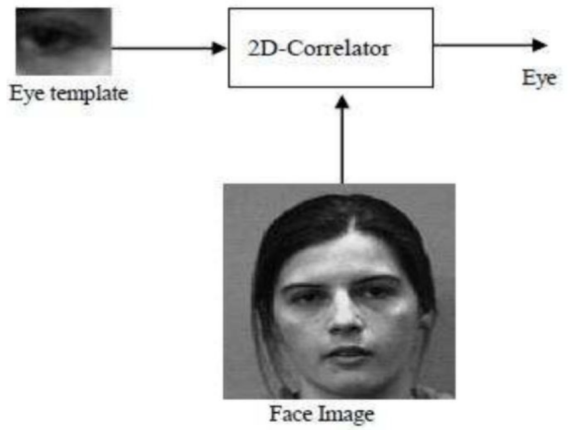


Abb. 3: Beispiel einer auf Vorlagen basierten Gesichtserkennung[BS14]

Zunächst wird also z.B. die Schablone eines Auges eingesetzt, um das Auge aus dem Bild zu erkennen. Anschließend wird ein Zusammenhang zwischen den Augenschablonen mit verschiedenen überlappenden Bereichen des Gesichtsbildes hergestellt. Die Region, die dann den maximalen Zusammenhang mit der Schablone aufweist, ist die Augenregion [BS14].

3.1.3 appearance-based

In Ansätzen dieser Gruppe wird das Bild als zweidimensionales Muster verarbeitet. Das Konzept der "Merkmale" ist in diesen Ansätzen ein anderes. Es unterscheidet sich von den einfachen Gesichtsmerkmalen wie Augen und Mund, und jede aus dem Bild extrahierte Eigenschaft wird als Merkmal bezeichnet [Hu09, BS14]. Die Gruppe von aussehensbasierten Ansätzen speichert wichtige Bildinformationen und weist überflüssige Informationen zurück, weshalb sie bei der Extraktion von Gesichtsmerkmalen die beste Leistung erzielt hat [BS14]. Bei diesen Verfahren werden Bildfilter angewandt. Diese werden entweder auf den gesamten Gesichtsbereich oder auf bestimmte Bereiche im Gesichtsbild eingesetzt, um einen Merkmalsvektor zu extrahieren [DS14].

3.1.4 color-based

Farbsegmentierungstechniken verwenden die Hautfarbe, um das Gesicht abzugrenzen [Hu09]. Die Hautregion wird unter der Verwendung von verschiedenen Farbmodellen, wie z.B. RGB, erfasst [BS14]. Alle Regionen innerhalb des Gesichts, die nicht die Farbe der Hautfarbe haben, gelten als Kandidaten für die Augen und den Mund. Die Leistungsfähigkeit beim Einsatz solcher Techniken in der Gesichtsbild-Datenbank ist eher begrenzt. Dies ist aufgrund der Vielfalt ethnischer Hintergründe der Fall [Hu09].

3.2 Gesichtsmerkmalsextraktion durch neuronale Netze

Nachdem im vorherigen Abschnitt die Hauptkategorien von Methoden zur Extraktion von Gesichtsmerkmalen vorgestellt wurden, soll im Folgenden ein weiterer, speziellerer Ansatz beschrieben werden, wie Gesichtsmerkmale extrahiert werden können. Bei diesem Ansatz handelt es sich um die Extraktion durch neuronale Netze.

Ansätze, die auf neuronalen Netzen basieren, sind aus Beispielbildern angelernt und stützen sich, um die relevanten Merkmale von Gesichtsbildern zu finden, auf Techniken des maschinellen Lernens [MM07].

Ein neuronales Netz (kurz NN) ist eine parallel-verteilte Informationsverarbeitung. Es ist für eine Datenverarbeitung geeignet, die mehrdeutige Elemente, wie z.B. Kanten einer

Nase und eines Mundes, enthält. Ein NN kann eine Regelmäßigkeit in einem Objekt finden, indem es sein Training wiederholt, da es eine Flexibilität aufweist, Koeffizienten zu ändern. Hierdurch wird eine Kreislauf-Struktur gebildet. Die Zwischenschicht eines hierarchischen NN sammelt Eingangsdaten, die durch Training erhalten werden, und führt durch wiederholtes Training eine Art multivariate Analyse, also eine Analyse die mehrere Variablen betrifft, durch. Wenn Back-Propagation-Training (BP) ideal durchgeführt wird, dann extrahiert das NN die k-dimensionalen Hauptkomponenten der Daten aus k "hidden units" [THT00].

In [THT00] werden z.B. die Merkmale von männlichen und weiblichen Gesichtern mit einem NN extrahiert. Verwendet wird hierfür ein NN, das aus einer dreischichtigen hierarchischen Struktur, einem "single hidden layer", besteht. Es wurden "hidden layers" erlangt, die bedeutend auf männliche und weibliche Gesichtsregionen reagieren. Durch die Analyse der Verbindungsgewichte zwischen der "hidden-layer unit" und der "input-layer unit" können die Gesichtsmerkmale extrahiert werden. Die Trainingsdaten des NN und die minimale Anzahl von "hidden layers" wurden durch einen genetischen Algorithmus erhalten. Für die gesamte Gesichtsregion und jede Gesichtskomponentenregion wurden mehrere unterschiedliche Anzahlen von Mosaikblöcken benutzt. Die Ergebnisse der Versuche geben, auf Grundlage von dem ganzen Gesicht und den einzelnen Regionen, folgendes wieder: Eigenschaften der Frisur und der Hautfarbe, die Größe der Augen und Augenbrauen, die Nasenhöhe und die Dicke der Lippen. Diese Ergebnisse waren denen, die durch andere psychologische Experimente und Gesichtsmessungen erhalten wurden, ähnlich. Das zeigt, dass ein NN Teilmerkmale von männlichen und weiblichen Gesichtern genau identifizieren kann [THT00].

In Bezug auf die Gesichtserkennung besteht der Vorteil der Verwendung neuronaler Netze darin, dass die Netze trainiert werden können und so mehr Wissen über die Variation von Gesichtsmustern gewinnen. Dadurch kann eine gute Verallgemeinerung erreicht werden [MM07].

4 Web-Service und Anwendungsbeispiel

Im folgenden Abschnitt soll ein Web-Service vorgestellt werden, der unterschiedliche Möglichkeiten für die Gesichtsentdeckung und -erkennung bietet und auch eine Beschreibung von Gesichtsmerkmalen liefert. Zudem soll ein Anwendungsbeispiel gegeben werden, in dem die Gesichtsmerkmalsextraktion angewendet wird.

4.1 Web-Service zur Gesichtsentdeckung und -erkennung

Betaface API [2] ist ein Web-Service für die Gesichtsentdeckung und -erkennung. Mit diesem Service können hochgeladene Bilddateien oder Bild-URLs gescannt, und Gesichter

[2] `https://www.betafaceapi.com/wpa/`

gefunden und analysiert werden. Ebenso werden durch die API auch Verifikations- und Identifizierungsdienste, wie ein Gesichtervergleich und die Gesichtersuche in einer eigenen Personendatenbank, angeboten. Die API kann folgendes liefern:

- Allgemeine Gesichtsinformationen,
- Klassifizierungen und
- erweiterte Messungen.

Zu den allgemeinen Gesichtsinformationen, die der Web-Service liefert, gehört die Erkennung mehrere Gesichter mit dessen Positionen, Größen und Winkel. Unter diesem Punkt werden auch die Stellen von 123 möglichen Gesichts-Landmarks und zugeschnittene Gesichtsbilder geliefert. Durch die Klassifizierung kann das Geschlecht, das Alter und die Volkszugehörigkeit abgeschätzt werden. Zudem kann beurteilt werden, ob der Ausdruck einer Person lächelnd oder neutral ist. Auch können Brillen, Schnurrbart und Bart durch den Web-Service erkannt werden. Unter die erweiterten Messungen fällt unter anderem die Beschreibung des Gesichts und der Gesichtsmerkmale durch die Form, relative Größe und Position. Die erweiterten Messungen betreffen auch die Farben der Augen, Haare, Haut, Kleidung und des Hintergrunds. Durch diese Messungen können auch Aussagen über die Menge der Gesichtsbehaarung und die ungefähre Frisur (Länge, Dicke und Form) getätigt werden.

Der Webservice kann für kommerzielle oder nichtkommerzielle Anwendungen genutzt werden. Durch die Funktionen der API kann eine Vielzahl von unterschiedlichen Attributen geliefert werden. Beispiele hierfür sind: spitze Nase, große Nase, Nasenbreite, ovales Gesicht, Augenabstand, schmale Augen, blasse Haut, hohe Wangenknochen, blondes Haar, starkes Make-up uvm. . Alle Angaben des Abschnitts 4.1 entstammen von der Betaface-Homepage [Be18].

4.2 Anwendungsbeispiel

Wie schon einleitend erwähnt, ist die Extraktion von Gesichtsmerkmalen ein wichtiger Prozess für die Gesichtsentdeckung, Ausdruckserkennung und Gesichtserkennung. Auch in der automatischen Anmerkung bzw. Beschriftung von Gesichtern (engl. “automatic face annotation“) wird die Gesichtsmerkmalsextraktion verwendet [PS17].

Die automatische Gesichtsbeschriftung ist eine Methode, die menschliche Gesichter in einen Bild identifiziert und diesen den entsprechenden Namen des Menschen zuweist. Eingesetzt wird diese Methode z.B. in den sozialen Netzwerken. Hierfür sind die Gesichtsentdeckung und -erkennung essentielle Aufgaben. Auch die Merkmalsextraktion ist in der automatischen Gesichtsbeschriftung für das Extrahieren der Merkmale von Gesichtern unverzichtbar [PS17]. Die Abbildung 4 zeigt den Ablauf der automatischen Gesichtsbeschriftung.

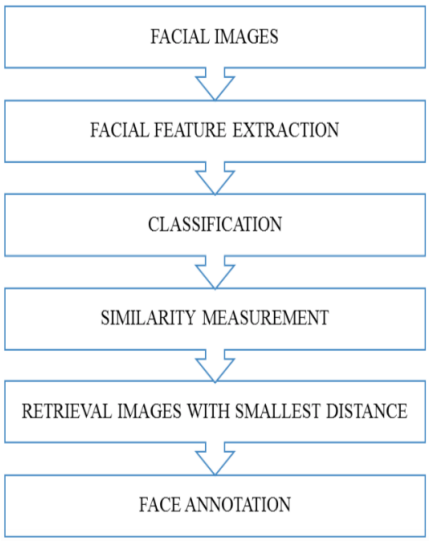


Abb. 4: Ablauf der automatischen Gesichtsbeschriftung [PS17]

Nachdem die Gesichtsmerkmale aus dem Gesichtsbild extrahiert wurden erfolgt die Klassifizierung. Bei dieser Bildklassifizierung werden die numerischen Eigenschaften verschiedener Bildmerkmale, in diesem Fall die Gesichtsmerkmale, analysiert und Daten in Kategorien organisiert. Anschließend erfolgt die Ähnlichkeitsmessung, um die Ähnlichkeit zwischen den Datenbankbildern und dem Abfragebild unter Verwendung der Merkmale zu finden. Im nächsten Schritt werden die Datenbankbilder nach aufsteigender Reihenfolge der Entfernung zum Abfragebild sortiert und abschließend entsprechend der Reihenfolge aus der Datenbank abgerufen. Die nächste Aufgabe ist die Gesichtsbeschriftung. Hierbei wird das Gesicht mit einer Bezeichnung versehen, die dem abgerufenen “nearest-neighbor“ zugeordnet ist. Das Abfragebild wird mit dieser Bezeichnung geteilt und die Gesichtsbeschriftungsaufgabe ist abgeschlossen [PS17].

5 Bezug zum Projekt PROPOSE.AI

In Bezug auf das Projekt PROPOSE.AI, in dem unter anderem ein Online-Brillenberater umgesetzt werden soll, kann die Extraktion von Gesichtsmerkmalen eingesetzt werden, um das Geben von Empfehlungen bei der Beratung von Brillen zu unterstützen.

Durch den Brillenberater sollen auf Basis eines fotografierten Gesichts Brillen empfohlen werden können. Hierbei soll es sich bei den Empfehlungen nicht um Brillen handeln, die vielleicht bei anderen Kunden beliebt sind oder momentan im Trend liegen, sondern um Brillen, die optimal zu dem Gesicht des Kunden passen. Hierfür können die Informationen, die aus der Extraktion von Gesichtsmerkmalen gewonnen werden, verwendet werden.

Der vorgestellte Web-Service aus Abschnitt 4.1 zeigt, wie viele Attribute aus einem hochgeladenen Bild erkannt werden können. So könnte bspw. eine Einschränkung der in Frage kommenden Brillen getroffen werden, indem z.B. die Kopfform oder die Form der Nase berücksichtigt wird. Je nach Kopfform können einige Brillenmodelle besser geeignet sein als andere. Auch der Nasenrücken könnte für die Empfehlung von einer Brille relevant sein, um einen angenehmen Tragekomfort und eine gute Passform zu gewährleisten. Neben diesen Punkten wäre es auch vorstellbar, eine Empfehlung hinsichtlich der Rahmenfarbe zu geben, indem bspw. die Haar-oder Augenfarbe hinzugezogen wird.

Literaturverzeichnis

[Be18] Betaface: , Betaface API. https://www.betafaceapi.com/wpa/, 2018. letzter Zugriff: 10.05.2018.

[BK16] Benedict, S. R.; Kumar, J. S.: Geometric shaped facial feature extraction for face recognition. In: 2016 IEEE International Conference on Advances in Computer Applications (ICACA). S. 275–278, Oct 2016.

[BR08] Bagherian, E.; Rahmat, R. W. O. K.: Facial feature extraction for face recognition: a review. In: 2008 International Symposium on Information Technology. Jgg. 2, S. 1–9, Aug 2008.

[BRU09] Bagherian, Elham; Rahmat, Rahmita Wirza; Udzir, Nur Izura: Extract of facial feature point. IJCSNS International Journal of Computer Science and Network Securtiy, 9, 2009.

[BS14] Bakshi, Urvashi; Singhal, Rohit: A survey on face detection methods and feature extraction techniques of face recognition. International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), 3:233–237, 2014.

[DS14] Dhall, Sonu; Sethi, Poonam: Geometric and appearance feature analysis for facial expression recognition. International Journal of Advanced Engineering Technology, V:1–11, Jul-Sept 2014.

[Hu09] Hung, Nguyen Viet: Facial Feature Extraction Based on Wavelet Transform. In (Deng, Hepu; Wang, Lanzhou; Wang, Fu Lee; Lei, Jingsheng, Hrsg.): Artificial Intelligence and Computational Intelligence. Springer Berlin Heidelberg, Berlin, Heidelberg, S. 330–339, 2009.

[MM07] Meethongjan, Kittikhun; Mohamad, Dzulkifli: A Summary of literature review: Face Recognition. 2007, Aug 2007.

[PS17] Patel, T.; Shah, B.: A survey on facial feature extraction techniques for automatic face annotation. In: 2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA). S. 224–228, Feb 2017.

[THT00] Tsuneo, Kanno; Hiroshi, Nagahashi; Takeshi, Agui: Extraction of male and female facial features using neural networks. Systems and Computers in Japan, 31(3):68–76, 2000.

[Wu12] Wu, Ying-Ming; Wang, Hsueh-Wu; Lu, Yen-Ling; Yen, Shin; Hsiao, Ying-Tung: Facial Feature Extraction and Applications: A Review. In (Pan, Jeng-Shyang; Chen, Shyi-Ming; Nguyen, Ngoc Thanh, Hrsg.): Intelligent Information and Database Systems. Springer Berlin Heidelberg, Berlin, Heidelberg, S. 228–238, 2012.

[YN02] Yen, G. G.; Nithianandan, N.: Facial feature extraction using genetic algorithm. In: Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on. Jgg. 2, S. 1895–1900, 2002.

Recurrent Neural Networks – Am Beispiel von Natural Language Processing

Jan-Hendrik Witte[1]

Abstract: Im Kern werden Recurrent Neural Networks dazu verwendet, um Gebrauch von sequentiellen Informationen zu machen. Aus diesem Grund eignen sich RNNs für viele unterschiedlichen Anwendungsfälle im Bereich des Natural Language Processing. Bei der Prognose des nächsten Wortes einer spezifischen Wörterkette bspw. ist die Beachtung sequentieller Informationen und deren zeitliche Abhängigkeit untereinander von elementarer Bedeutung. In dieser Arbeit sollen die Potentiale von RNNs im Zusammenhang mit NLP verdeutlicht und eine Übersicht verschiedener Anwendungsfälle erarbeitet werden. Es wird ebenfalls auf potentielle Limitationen von RNNs in spezifischen Anwendungsfällen eingegangen und wie diese gelöst werden können. Ausgehend von den jeweiligen Limitationen soll zudem eine Übersicht unterschiedlicher RNN-Architekturen in Verbindung mit NLP-Aufgaben geschaffen werden. Abschließend wird eine Verknüpfung des Themas mit den Anforderungen des Projektes hergestellt, um den Bezug auf die jeweiligen Anwendungsfälle herzustellen.

1 Grundlagen

1.1 Neuronale Netze

Aufgrund der vielen unterschiedlichen Netzwerktypen ist es schwierig, eine allgemein gültige Definition für den Begriff der neuronalen Netzte zu finden [RW11]. Allerdings weisen neuronale Netze Gemeinsamkeiten im Aufbau und ihrer Funktionsweise auf. Die Gemeinsamkeiten, welche neuronale Netzwerke miteinander verbindet, sind die Kernfunktionsweisen. Diese wären das Aufnehmen, Verarbeiten und Ausgeben von Informationen und deren nachfolgenden Umstrukturierungen in einem zyklischen Verarbeitungsprozess [RW11]. Was dies genau bedeutet, kann durch die Beschreibung der einzelnen Komponenten eines neuronalen Netzes und deren Arbeitsweise verdeutlicht werden.

Neuronale Netze setzten sich aus einer endlichen Menge von Neuronen bzw. Units zusammen. Es existieren drei unterschiedliche Typen von Units, welche jeweils unterschiedliche Funktionen im Netzwerk erfüllen. Input-Units nehmen eingehende Signale in Form von Zahlen auf. Output-Units geben Signale in Form von Zahlen an die Außenwelt weiter. Hidden-Units sind Units, welche sich zwischen Input- und Output-Units befinden [RW11].

[1] Carl von Ossietzky Universität, VLBA, Ammerländer Heerstraße 114, 26129 Oldenburg, jan-hendrik.witte@uni-oldenburg.de

Dabei ist anzumerken, dass die jeweiligen Mengen der Input- sowie Output-Units nicht disjunkt sein müssen und somit beide Funktionen in einer Unit vereinen kann [Kr15].

Vertikal angeordnete Units werden als Layer zusammengefasst, wobei übereinander angeordnete Hidden-Units als Hidden-Layer bezeichnet werden. Im Normalfall besteht ein neuronales Netz aus genau einem Input-Layer und einem Output-Layer. Die Anzahl der verwendeten Hidden-Layers variiert und kann je Anwendungsfall sowohl aus null, einer oder mehrere Schichten bestehen [RW11]. Units sind über eine endliche Menge von Kanten miteinander verbunden [Kr15]. Der Einfluss einer solchen Verbindung zwischen Units wird mithilfe eines Gewichts ausgedrückt. Je höher der Absolutbetrag des jeweiligen Gewichts ist, desto stärker ist sein Einfluss auf die jeweilige Unit. Gewichte sind ein elementarer Bestandteil des Netzwerkes, in ihnen ist im übertragenem Sinne das Wissen des neuronalen Netzes gespeichert. Der iterative Veränderungsprozess der Gewichtsbeträge der Kanten zwischen den Units stellt dabei den Lernprozess des Netzwerkes dar [Kr15].

Mit Ausnahme von Input- und Bias-Units [RW11], sind einer Unit drei zentrale Zustandsgrößen zugeordnet. Die Netzeingabe bzw. Netzinput, die Aktivierungsfunktion und die Ausgabefunktion [Kr15]. Der Netzinput einer Unit setzt sich allen eingehenden Inputs dieser Unit zusammen [Tr03]. Die Propagierungsfunktion bestimmt dabei, wie sich der Netzinput der jeweiligen Unit ermittelt. Häufig wird für die Propagierungsfunktion die Linearkombination verwendet, sodass die Gewichtete Summe der jeweiligen Inputs verwendet wird [RW11]. Anhand des Netzinputs kann unter Zunahme der Aktivierungsfunktion das Aktivitätslevel der jeweiligen Unit bestimmt werden [RW11]. Ist mit dieser Summe ein gesetzter Schwellenwert überschritten, kann ein Signal an anbindende Units weitergegeben werden. Aktivierungsfunktionen werden i.d.R. durch eine Sigmoid- oder Tangens-hyperbolicus-Funktion abgebildet [Tr03]. Das Besondere an diesen Aktivierungsfunktionen ist, dass sie, anders als bspw. binären Funktionen, an allen Stellen differenzierbar sind. Dies ist eine elementare Voraussetzung zur Anwendung des Gradientenabstiegsverfahrens, welches im späteren Verlauf dieser Arbeit erneut aufgegriffen wird [RW11]. Die dritte und letzte Zustandsgröße, die Ausgabefunktion bzw. der Output einer Unit, ist gleichzusetzten mit dem aus der Aktivierungsfunktion ermittelten Aktivitätslevel [RW11].

Ein neuronales Netz lässt sich in Form einer Matrix abbilden. Dies hat den Vorteil, dass die anfallenden Berechnungen im Netzwerk wesentlich einfacher und zusammenfassender vorgenommen werden können. Diese Matrix wird auch als Gewichtsmatrix bezeichnet, da die Zahlenwerte der Gewichte der jeweiligen Katen die Werte in der Matrix repräsentieren. Sofern keine Hidden-Layers vorhanden sind, kann ein neuronales Netzwerk durch eine Gewichtsmatrix dargestellt werden. Existieren allerdings Hidden-Layers, ist die Anzahl der benötigten Gewichtsmatrizen äquivalent zur Anzahl der vorhandenen Hidden-Layers [RW11].

1.2 Natural Language Processing

Natural Language Processing kann in zwei Kerndisziplinen unterteilt werden. Zum einen geht es darum, der Maschine bzw. dem Computer die Konzepte natürlicher Sprachen zu lehren. Zum anderen geht es darum, die Maschine bzw. den Computer dazu zu bringen, dieses Wissen im Rahmen spezifischer Anwendungsfälle umsetzbar zu machen [Ch03]. Natural Language Processing kann in viele unterschiedliche Teildisziplinen differenziert werden, bspw. Machine Translation, Natural Language Text Processing, Speech Recognition oder Articifial Intelligence [Ch03]. Auch diese Disziplinen können nochmals kategorisiert werden, weshalb an dieser Stelle eine Eingrenzung des Themengebiets des Natural Language Processing vorgenommen wird. Folglich wird versucht, Bezug auf die im Rahmen der Projektgruppe skizzierten Anwendungsfälle der Brille24 zu nehmen. Natural Language Processing könnte dementsprechend interessant für die Trenderkennung in sozialen Medien sein. Vor allem jedoch geht es beim Natural Language Processing um die Verarbeitung sequentieller Datenketten. Machine Translation oder Speech Recognition stellen jeweils Anwendungsfälle dar, in denen Sequenzen von Daten betrachtet werden und diese somit räumliche als auch zeitliche Abhängigkeiten untereinander aufweisen [LBE15]. Im Folgenden soll zunächst aufgezeigt werden, was rekurrente Netzwerke von traditionellen neuronalen Netzwerken unterscheidet, warum sich diese für Anwendungsfälle im NLP eignen.

2 Recurrent Neural Network

In klassischen neuronalen Netzwerken wird angenommen, dass sämtliche Inputs bzw. Outputs unabhängig voneinander sind [Br15a]. Diese Annahme resultiert darin, das nach der Betrachtung eines Datenpunkts oder nach dem Abschluss einer Iteration alle zuvor gesammelten Informationen zum Zustand des Netzwerkes verloren gehen [LBE15]. In Anwendungsfällen, in denen Sequenzen betrachtet werden und betrachtete Datenpunkte zeitlich und räumlich miteinander zusammenhängen, ist die Annahme der Unabhängigkeit von Units allerdings problematisch. Betrachtet man bspw. Szenarien, in denen einzelne Frames eines Videoausschnittes, Abschnitte einer Audioaufnahme oder einzelne Wörter in einem Satz betrachtet werden, versagt die Unabhängigkeitsannahme [LBE15]. Bei dem Versuch, ein Netz zu trainieren, welches effektiv fundierte Prognosen über zukünftige Zustände geben soll, sind vorhergegangene Zustände von signifikanter Bedeutung [Br15a]. Die Problematik der Unabhängigkeitsannahme kann mit folgendem fiktiven Beispiel verdeutlicht werden.

Eine Person steht vor dem Pariser Eifelturm. Durch logische Verknüpfung der gesammelten Inputs kommt die Person zu der Schlussfolgerung, dass sie sich in Frankreich befinden muss. Dieselbe Person geht einige Zeit durch Paris und bemerkt eine chinesische Familie in einer kargen Seitenstraße. Zunächst werden die von der Außenwelt eintreffenden Signale verarbeitet.

In diesem Zustand wirken anderen Reize mit unterschiedlichen Gewichtungen auf die Person ein, unter anderem die unmittelbar vor der Person befindlichen chinesischen Familie. Ebenso wie im vorherigen Zustand kann durch das eintreffen neuer Signale differenzierte Schlussfolgerungen getroffen werden. In diesem Fall könnte die Person annehmen, dass sie sich gerade in China befindet. Da die Person jedoch wenige Minuten zuvor noch den Eifelturm betrachtet hat, kann rückblickend auf die zuvor geschehenen Ereignisse die Schlussfolgerung gezogen werden, dass die Person sich definitiv noch in Frankreich befinden muss. Die Maschine, welche mit einem klassischen neuronalen Netzwerk trainiert wird bzw. wurde, kann auf diese Form der Schlussfolgerung nicht zurückgreifen. Relevant sind nur die aktuell unmittelbaren Inputs und Signale des jeweiligen Zustands. Für die Maschine könnte dies genauso gut bedeuten, dass sie sich gerade in China befindet, da sie keine Möglichkeit besitzt, auf die Informationen der zuvor berechneten Zustände zurückzugreifen.

Ähnliche Problematik von neuronalen Netzen lässt sich auch auf Themengebiete des Natural Language Processing übertragen. Möchte man das Nächste Wort in einem Satz vorhersagen, müssen die Informationen der zuvor aufgeführten Wörter beachtet werden [Br15a]. Diese elementaren Problemstellungen können mithilfe von Recurrent Neural Networks, kurz RNNs, adressiert werden. Der Kerngedanken hinter RNNs ist es, Gebrauch von sequentiellen Informationen in Daten zu machen [Br15a]. Sie werden als „recurrent“ bezeichnet, da sie für jedes Element einer Sequenz bei der Verrechnung der Informationen die Zustände vergangener Iterationen mit einbezieht. Im übertragenen Sinne verfügt das Netzwerk somit eine Art Gedächtnis, in welchem die Informationen bereits vergangener Berechnungen erfasst werden können [Br15a]. Wie genau dies ermöglicht wird, kann durch folgende Grafik verdeutlicht werden.

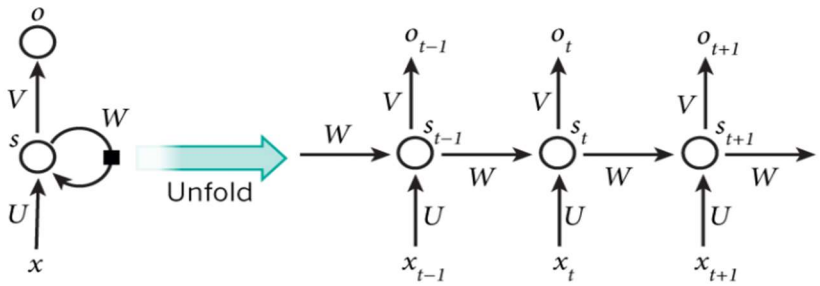


Abb. 1: Recurrent Neural Network

In Abb. 1 werden zwei Sachen veranschaulicht.

Auf der linken Seite wird zum einen das einfache Modell eines RNN abgebildet.

- x stellt den Input an einem gegeben Zeitpunkt t dar.
- s stellt den „Hidden-State“ des Netzwerkes zu einem Zeitpunkt t dar. Dieser wird in einem Wert ausgedrückt und setzt sich zusammen aus den Berechnungen vergangener Hidden-States und dem aktuellen Inputwert
- o ist der Output zu einem Zeitpunkt t [Br15a].

Wie im Beispiel der Grafik veranschaulicht, referenziert die Unit beim Durchlauf auf sich Selbst. Bei dieser Form der Rückkopplung spricht man von einer direkten Rückkopplung. Es existieren allerdings noch weitere Formen der Rückkopplung zwischen Units. Bei indirekten Rückkopplungen bspw. referenziert das aktuell betrachtete Objekt zurück auf die vorhergegangene Unit.

Somit werden die Aktivitäten und die damit einhergehenden Werte an vorherige Units zurückgegeben. Durch die seitliche Rückkopplung wird ermöglicht, Informationen an Units derselben Layer weiterzugeben [RW11].

Auf der rechten Seite wird der Entfaltungsprozess des Netzwerkes nach Parameterübergabe verdeutlicht. Bei der Betrachtung von sequentiellen Daten, bspw. bei einer Wörterkette mit fünf Wörtern, würde sich das Netzwerk in ein fünfschichtiges Netzwerk entfalten [Br15a]. Der Wert s zum Zeitpunkt t verkörpert dabei das Gedächtnis oder die Erinnerungen des Netzwerkes und erfasst die Zustände und Berechnungen vorheriger Iterationen. Der Output o zu einem Zeitpunkt t bezieht sich dementsprechend immer auf die Berechnungen vorheriger Units [Br15a], sodass Teilinformationen aus sämtlichen vorangegangenen Zeitpunkte erfasst werden können [RW11]. Die generelle Funktionsweise eines RNNs wurde somit verdeutlicht. Nun kann anschließend auf die jeweiligen Anwendungsfälle von RNNs im Zusammenhang mit NLP eingegangen werden.

2.1 Anwendung von RNNs im Bereich NLP

Einige der Kerngebiete des Natural Language Processing wurden bereits im Grundlagenkapitel genannt. Wie bereits angedeutet, wird innerhalb dieser Arbeit der Fokus auf Themenbereiche gelegt, welche im Rahmen der Projektgruppe von Relevanz sein könnten. Diese umfassen bspw. Spracherkennung, maschinelle Übersetzung, Textgenerierung und Sprachmodellierung. In der Literatur existieren viele Anwendungsfälle, in denen Recurrent Neural Networks verwendet wurde, um sich genau diesen genannten Themengebieten zu widmen. In [Zh18] wurden RNNs verwendet, ein Spracherkennungssystem zu entwickeln, welches Audiodaten ohne zwischengeschaltete phonetische Abschrift der Audioaufnahme direkt in Textform umwandeln kann.

Mithilfe der Kombination aus einem Recurrent Neural Newtork und einem Recursive Neural Network wird in [Li14] versucht, ein Modell zur direkten und unmittelbaren maschinellen Übersetzung übergebener Sprache entwickelt.

In [SME11] wird mithilfe einer angepassten Form eines RNNs aufgezeigt, wie diese effektiv für die Verarbeitung von sequentiellen Daten trainiert werden können. Ziel war es, die Wirksamkeit und Fähigkeit umfangreicher RNNs in der Prognose des nächsten Wortes in einer Reihe von Wörtern aufzuzeigen.

In einigen der aufgezeigten Paper wird eine besondere Form von RNNs verwendet, sogenannte LSTM-Netzwerke bzw. Long-Term-Short-Memory-Netzwerke. Diese werden allerdings erst im weiterführenden Kapitel erläutert, da diese in enger Verbindung zum bereits aufgeführten Gradientenabstiegsverfahren stehen und zentrale Problemstellen traditioneller RNNs adressieren.

Ein ebenfalls oft gefallener Begriff im Rahmen der aufgeführten Veröffentlichungen ist der des „Language Models“ bzw. Sprachmodells. Für viele der zuvor genannten Disziplinen des NLP stellen Sprachmodelle eine essentielle Grundlage dar, um sequentielle Daten zu verarbeiten. Sprachmodelle dienen dazu, die Wahrscheinlichkeiten für bestimmte Wörterketten bzw. Sätze in einer gegebenen Sprache zu prognostizieren. Durch die Gliederung des Satzes in einzelne Sequenzen mit jeweils eigenen spezifischen Eintrittswahrscheinlichkeiten für jedes Wort der Kette, kann die gesamte Eintrittswahrscheinlichkeit ermittelt werden [CFL13]. Lange wurden n-gram-Modelle zur Generierung statistischer Sprachmodelle herangezogen. Größte Kritik an jeweiligen Modellen ist jedoch, dass kein wirkliches „Lernen“ bzw. „Verstehen“ in den Modellen stattfindet. Ursache dafür sind die Limitationen, welche n-gram-Modelle mit sich bringen. Die Prognostizierung zukünftiger Wörter wird lediglich in Abhängigkeit der zwei zuvor betrachteten Wörter ermittelt, was bei der Verarbeitung von sequentieller Datenreihen ungenügend ist [Mi11]. In [Mi10] und weiterführend in [Mi11] wurde sich mit der Generierung eines Sprachmodells auf Basis von RNNs beschäftigt. Verwendet wurde dazu ein Simple Reccurent Neural Network bzw. Elman Network, welches in [El90] weiter beschrieben wird. Ergebnisse der Studie ergaben, dass RNNs die damaligen State oft he Art-Lösungen maßgeblich übertreffen, selbst wenn diese mit wesentlich weniger Daten trainiert worden sind als der Gegenpart. Des Weiteren konnte die Error-Rate des generierten Modells im, bei gleicher Menge zugefügter Daten, im Vergleich um 18% reduziert werden. Damit wurde widerlegt, dass es zum einen nicht nur auf das Zählen von n-grams bei der Erstellung von Sprachmodellen ankommt und zum anderen, dass zur Verbesserung der Modelle nicht nur die Menge an zugeführten Daten entscheidend ist. Allerdings kamen die Autoren ebenfalls zu dem Ergebnis, dass Simple RNNs Probleme bei der Erfassung von Abhängigkeiten über längere Zeiträume aufweisen. Dies schafft die Überleitung zum nächsten Unterkapitel, indem das Problem der Erfassung von Abhängigkeiten über längere Zeiträume unter Verwendung des Gradientenabstiegverfahren in Kombination spezifischer Trainingsalgorithmen bei RNNs aufgezeigt werden soll.

3 Gradientenabstiegsverfahren

Im Grundlagenkapitel wurde bereits angedeutet, dass die iterative und zyklische Veränderung der Gewichte der Kanten zwischen Units den Lernprozess im Netzwerk darstellt. Um das Netzwerk zu trainieren, werden Lernverfahren angewendet, wie bspw. das bereits genannte Gradientenabstiegsverfahren.

Beim Trainieren eines Netzwerkes sind die gewünschten Ausgabeparameter vorgegeben. Ein Netz gilt dann als trainiert, wenn die Ausgaben der aktuellen Iteration mit den Ausgaben des angestrebten Zustandes übereinstimmen [Beo. J.]. Es wird also nach einer Kombination von Gewichtswerte der Kanten gesucht, welche als Ergebnis den vorgegebenen Parametern entspricht.

Eine Möglichkeit, um diese Kombination zu finden, wäre, zu allen Kombinationen von Gewichten einen Fehlerterm zu finden. Die Kombination mit dem geringsten Fehlerterm wäre dann die optimalste Lösung und hinsichtlich des Fehlers das absolute Minimum [RW11]. Was für einen zweidimensionalen Raum mit nur einem Gewicht durchaus vorstellbar wäre, gestaltet sich diese Herangehensweise bei der Betrachtung n-dimensionaler Räume deutlich komplizierter. Würde man sämtliche Fehlerterme in einem Koordinatensystem abbilden, so entsteht eine Fehlerkurve, welche einer Gebirgslandschaft ähnelt. Der Rechenaufwand, um zunächst sämtliche Fehlerterme sowie anschließend das lokale Minimum zu berechnen, wäre viel zu aufwändig [RW11].

Um dieses Problem zu umgehen und dennoch den Punkt des lokalen Minimums der Fehlerfunktion zu finden, verwendet man das Gradientenabstiegsverfahren. Zunächst wird der Gradient bzw. die Steigung an einem zufällig gewählten Punkt der Fehlerfunktion bestimmt. Indem anschließend einzelne Gewichte der Kanten entgegen des Gefälles der Fehlerfunktion justiert werden, kann mithilfe der partiellen Ableitung ein neuer Punkt auf der Fehlerkurve ermittelt werden. Dieser iterative Prozess wird solange durchgeführt, bis das lokale Minimum erreicht wurde [RW11].

Das Gradientenabstiegsverfahren ist die Grundlage für viele unterschiedliche Trainingsalgorithmen. Wenn es darum geht, aktuelle Outputs des Netzwerkes an einen bestimmten Zieloutput anzunähern, wird häufig zur Methode der Backpropagation auf Basis des Gradientenabstiegsverfahrens gegriffen. Zum Erkennen von Mustern innerhalb sequentieller Datenobjekte hat sich zudem eine weitere Abwandlung der Backpropagation-Methode entwickelt, Backpropagation Though Time (BPTT) [We90]. In [We90, Br15b] können alle nötigen Informationen zur Funktionsweise und Notation von Backpropagation-, insbesondere BPTT-Methoden, nachgelesen werden. Die Kombination von RNNs und BPTT ist eine gängige Methode, Netze auf Basis sequentieller Daten zu trainieren, da die einzelnen Gradienten jeder Iteration vergleichsweise einfach zu berechnen sind [SME11]. Es wurde bereits angedeutet, dass traditionelle RNNs Probleme bei der Erfassung langfristiger Abhängigkeiten in sequentiellen Datenketten aufweisen. Dieses Problem soll im nachfolgendem Kapitel näher betrachtet werden.

3.1 Gradientenproblem

In [Ho91] und weiterführend in [BSF94] wurde zum ersten Mal bewiesen, dass beim Trainieren eines RNN auf Basis von BPTT die jeweilig berechneten Gradienten pro Iteration “schwinden” bzw. “explodieren“. Auf Grundlage dieser entwickelten Konzepte, das sogenannte Vanishing-Gradient-Problem bzw. Exploding-Gradient-Problem argumentierten die Autoren, dass RNNs keine langfristigen zeitlichen Abhängigkeiten lernen kann, wenn das Gradientenverfahren als Trainingsmethode verwendet wird. In [PMB12] wird näher erläutert, wie es beim Berechnen der Gradienten zu den oben genannten Problemen kommt.

Bezogen auf das Vanishing-Gradient-Problem liegt das Problem in der Aktivierungsfunktion. Da beim Gradientenverfahren stets die partielle Ableitung an einem bestimmten Punkt der Fehlerfunktion berechnet werden muss, müssen Aktivierungsfunktionen gewählt werden, welche an jedem Punkt ableitbar sind. Aus diesem Grund wird die Aktivierungsfunktion oftmals mithilfe einer Sigmoid- oder Tangens-Hyperbolicus-Funktion abgebildet. Beide Funktionen haben einen abgeleiteten Wert von null an beiden Enden. Neuronen werden als gesättigt bezeichnet, wenn der berechnete Gradient des jeweiligen Zustandes null entspricht. Ist dieser Punkt erreicht, wird eine Art Kettenreaktion ausgelöst, welche andere Gradienten aus vorherigen Schichten ebenfalls gen null lenken. Die mathematische Ursache dieses Phänomens liegt darin, dass die Werte in den jeweiligen Matrizen durch den sinkenden Gradienten ebenfalls sinken, sodass bei der Multiplikation der Matrizen vorheriger Schichten der Wert des Gradienten exponentiell sinkt und nach wenigen Iterationen nahezu verschwunden ist [Br15b]. Daraus resultiert, dass Kontributionen der Gradienten aus weiter entfernten Schichten ebenfalls verschwinden und somit keinen Einfluss mehr auf aktuelle Zustände des Netzes nehmen [Br15b]. Das Netzwerk „vergisst“ und hat zur Folge, dass keine längerfristigen Abhängigkeiten im Netzwerk berücksichtig werden können und somit bei der Betrachtung größerer sequentieller Datensätze ungeeignet ist [BSF94]. Dies bedeutet jedoch nicht, dass RNNs gänzlich ungeeignet für das Trainieren auf bestimmte Anwendungsfälle sind. Es bedeutet lediglich, dass das Gradientenabstiegsverfahren zunehmend an effizient verliert, je größer die zeitliche Spannweite der Abhängigkeiten im Netzwerk wird [BSF94].

Auf das Problem des Exploding-Gradients wird an dieser Stelle nicht weiter eingegangen, da es ein wesentlich unkompliziertes Problem im Vergleich zum Vanishing-Gradient darstellt. Nähere Informationen zu diesem Thema kann in [BSF94, PMB12] gefunden werden. Im weiteren Verlauf sollen potentielle Lösungsansätze für die genannten Probleme aufgeführt werden.

4 Long Short-term Memory

Eine Lösung, das Problem des Vanishing-Gradients zu umgehen, wurde erstmals in [HS97] vorgestellt. Um längerfristige Abhängigkeiten im Modell erfassen zu können, wurde das klassische RNN-Modell modifiziert und um sogenannten Memory Cells erweitert. Mithilfe dieser Memory- bzw. LSTM-Zellen wird ermöglicht, dass die jeweiligen Werte der Berechnungen der Fehler im Gradientenabstiegsverfahren konstanter gehalten werden können [HS97]. Der Aufbau und die Funktionsweise einer Memory-Cell kann in [HS97] ebenfalls eingesehen werden. Für ein besseres Verständnis darüber, wie eine solche Zelle die jeweiligen Zustände der Hidden-Layers berechnet, kann in [Ol15] nachgeschaut werden. Ebenso wie RNNs lassen sich LSTMs ebenfalls in vielen Anwendungsfällen des NLPs einsetzen. In der Literatur und Forschung existieren viele unterschiedliche Ansätze und Variationen, LSTMs zu verwenden und einzusetzen. An dieser Stelle soll weniger auf die allgemeine Funktionsweise von LSTMs eingegangen werden, sondern ein Literaturüberblick gegeben werden, wie LSTMs in der Forschung verwendet werden. In [Za16] wird ein durch LSTM-Zellen modifiziertes RNN zur automatischen Spracherkennung eingesetzt. Sprachidentifikation strebt danach, auf Basis übergebener Abschnitte von Sprachäußerungen automatisch die jeweilige Sprache zu identifizieren. Ziel des Papers ist es, dass Netzwerk dahingehend zu trainieren, dass es selbst bei der Übergabe sehr kurzer Sprachäußerungen automatisch die richtige Sprache identifiziert. Ein weiteres Themengebiet, in welchem RNNs in Kombination mit LSTMs eingesetzt werden, ist das der Natural Language Inference. NLI beschäftigt sich mit der Problematik, ob und wie auf Grundlage eines Prämissensatzes P ein Hypothesensatz H abgeleitet werden kann [WJ16]. Folglich geht es darum, auf Basis übergebener Datensätze fundierte Schlussfolgerungen abzuleiten und Hypothesen zu bilden. NLI ist ein wichtiger Bestandteil zur Behandlung spezifischer Anwendungsfälle wie bspw. bei semantischen Analysen, Textzusammenfassung und bei automatischer Fragebeantwortung [WJ16]. In [WJ16] wird eine spezielle und modifizierte LSTM-Architektur vorgestellt, welches speziell auf NLI-Anwendungsfälle entwickelt wurde. Eine weitere interessante Quelle beschäftigt sich mit dem Einsatz von RNNs im Zusammenhang mit Sentimentenanalysen [Ar17]. Das erst kürzlich vorgestellte Konzept der Layer-wise Relevance Propagation (LRP), welches zuvor nur in klassischen feed-forward-Netzen angewandt wurde, wird im Rahmen dieses Papers ebenfalls auf RNNs übertragen. LRP erwies sich im Vergleich zu gradientenbasierten Techniken als effizientere Methode, wenn es darum ging, Erklärungen zu generieren. Die LRP-Konzepte wurden mit spezifischen Regeln erweitert, sodass multiplikative Interaktionen in LSTM-Zellen gehandhabt werden können und sich somit besser für die Modellierung weitreichender Interaktionen in Texten eignet [Ar17]. Eine weiter interessante Quelle zum Thema Sentimentenanalyse kann in [BA16] gefunden werden. Für tiefergehende Informationen zur Performance unterschiedlicher LSTM-Modelle und –Architekturen kann in [Br15c] eine Gegenüberstellung und anschließende Evaluation verschiedener LSTM-Modelle eingesehen werden.

5 Schlussfolgerungen für das Projekt

RNNs eignen sich für viele der aktuellen und bedeutenden Aufgabenfelder des NLPs. Zu beachten ist dabei, inwiefern die Abhängigkeiten über Zeiträume bei der Betrachtung sequentieller Datensätze von Relevanz sind. Je größer die zeitlichen und räumlichen Abhängigkeiten zwischen Datensätzen ist, desto ineffizienter werden traditionelle RNNs, weshalb adaptierte RNN-Architekturen wie LSTMs zurückgegriffen werden muss. Dies ist in Anbetracht des Projektes zu berücksichtigen. Ein angestrebtes Ziel des Projekts ist die intelligente Trenderkennung in sozialen Netzwerken, um Beschaffungsprozesse zu optimieren. Im Rahmen dieser Arbeit wurden Paper vorgestellt, welche auf den genannten Anwendungsfall übertragbar sind. NLI und die Sentimentanalyse, vor allem jene in Bezug auf Twitter, auf Basis einer RNN-LSTM-Architektur könnten diesbezüglich interessant sein. Es ist jedoch zu differenzieren, ob für unseren Anwendungsfall ein Lösungsansatz auf Basis von LSTMs überhaupt notwendig ist, oder ob dies das Projekt nur unnötig verkompliziert. Es existieren noch viele andere unterschiedliche Lösungsansätze, um das Vanishing- bzw. Exploding-Gradient-Problem zu umgehen, welche weniger komplex sind als LSTMs. Sogenannte Gated Recurrent Units (GRU), vorgestellt in [Ch14], stellen im Kern eine vereinfachte Version von LSTMs dar. Aus Platzmangel konnte allerdings nicht weiter auf dieses Konzept eingegangen werden, da aktuell LSTMs die weitaus verbreitetere Architektur im Umgang mit dem Gradientenproblem ist.

Literaturverzeichnis

Ar17	Arras, L. et al.: Explaining Recurrent Neural Network Predictions in Sentiment Analysis, 2017.
BA16	Balikas, G.; Amini, M.-R.: TwiSE at SemEval-2016 Task 4: Twitter Sentiment Classification, 2016.
Beo. J.	Behr, T.: Bachpropagation. http://www.thomas-behr.de/studium/neuronale_netze/Neuronale_Netze.html.
Br15a	Britz, D.: Recurrent Neural Networks Tutorial. http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/, 02.05.2018.
Br15b	Britz, D.: Recurrent Neural Networks Tutorial, Part 3 – Backpropagation Through Time and Vanishing Gradients. http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/.
Br15c	Breuel, T. M.: Benchmarking of LSTM Networks, 2015.
BSF94	Bengio, Y.; Simard, P.; Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. In IEEE transactions on neural networks, 1994, 5; S. 157–166.
CFL13	Clark, A.; Fox, C.; Lappin, S. Hrsg.: The handbook of computational linguistics and natural language processing. Wiley-Blackwell, Chichester, 2013.
Ch03	Chowdhury, G. G.: Natural language processing. In Annual Review of Information Science and Technology, 2003, 37; S. 51–89.
Ch14	Cho, K. et al.: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, 2014.
El90	Elman, J. L.: Finding Structure in Time. In Cognitive Science, 1990, 14; S. 179–211.
Ho91	Hochreiter, S.: Untersuchungen zu dynamischen neuronalen Netzen, 1991.
HS97	Hochreiter, S.; Schmidhuber, J.: Long Short-Term Memory. In Neural Computation, 1997, 9; S. 1735–1780.
Kr15	Kruse, R. et al.: Computational Intelligence. Eine methodische Einführung in künstliche neuronale Netze, evolutionäre Algorithmen, Fuzzy-Systeme und Bayes-Netze. Springer Vieweg, Wiesbaden, 2015.
LBE15	Lipton, Z. C.; Berkowitz, J.; Elkan, C.: A Critical Review of Recurrent Neural Networks for Sequence Learning, 2015.

Li14	Liu, S. et al.: A Recursive Recurrent Neural Network for Statistical Machine Translation. In 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference, 2014, 1.
Mi	Mikolov, T. et al.: Extensions of recurrent neural network language model: 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2011 - 2011; S. 5528–5531.
Mi10	Mikolov, T.; Karafiát, M.; Burget, L.; Cernocký, J.; Khudanpur, S. Mikolov, T. et al.: Recurrent neural network based language model, 2010.
Ol15	Olah, C.: Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.
PMB12	Pascanu, R.; Mikolov, T.; Bengio, Y.: On the difficulty of training Recurrent Neural Networks, 2012.
RW11	Rey, G. D.; Wender, K. F.: Neuronale Netze. Eine Einführung in die Grundlagen, Anwendungen und Datenauswertung. Huber, Bern, 2011.
SME11	Sutskever, I.; Martens, J.; E. Hinton, G.: Generating Text with Recurrent Neural Networks. In Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011.
Tr03	Traeger, M. et al.: Künstliche neuronale Netze. In Der Anaesthesist, 2003, 52; S. 1055–1061.
We90	Werbos, P. J.: Backpropagation through time: what it does and how to do it. In Proceedings of the IEEE, 1990, 78; S. 1550–1560.
WJ16	Wang, S.; Jiang, J.: Learning Natural Language Inference with LSTM, 2016.
Za16	Zazo, R. et al.: Language Identification in Short Utterances Using Long Short-Term Memory (LSTM) Recurrent Neural Networks. In PloS one, 2016, 11; e0146917.
Zh18	Zhang, Y. et al.: Towards End-to-End Speech Recognition with Deep Convolutional Neural Networks, 2018.

Neuro-Fuzzy-Systeme

Hybride Systeme

Stefan Zurborg[1]

Abstract: Künstliche neuronale Netze haben sich in den letzten Jahren zu einem Standardwerkzeug in dem Gebiet der künstlichen Intelligenz entwickelt. Durch verschiedene Lernalgorithmen können sie beispielsweise Muster erkennen und Verfahren des maschinellen Lernens unterstützen. Allerdings bestehen auch Nachteile und Probleme bezüglich dieser Technologie. Zum Beispiel stellen die neuronalen Netze für den Nutzer eine Black Box dar und es kann kaum Regelwissen aus den trainierten Systemen extrahiert werden. Aus diesem Grund wird der zusätzliche Einsatz von Fuzzy-Logik in sogenannten Neuro-Fuzzy-Systemen diskutiert. Im Rahmen dieser Arbeit werden speziell die hybriden Systeme anhand von ANFIS und NEFCON vorgestellt.

Keywords: Neuro-Fuzzy-Systeme; Hybride Systeme; ANFIS; NEFCON

1 Einleitung

Sowohl künstliche neuronale Netze als auch Fuzzy-Systeme versuchen das Verhalten von Experten nachzubilden, mit dem Ziel komplexe Problemstellungen zu lösen. Der Einsatzbereich reicht von der Regelung dynamischer Systeme bis zur Beurteilung von Entscheidungsvorgängen. Trotz der oberflächlichen Gemeinsamkeiten bestehen eindeutige Unterschiede zwischen den beiden genannten Techniken. Beispielsweise bieten die neuronalen Netze Vorteile hinsichtlich der Lernfähigkeit, während die exakte Interpretierbarkeit nur bei den Fuzzy-Systemen gegeben ist. Neuro-Fuzzy-Systeme sollen, durch eine Kopplung beider Vorgehensweisen, deren Vorzüge kombinieren [Bo03].

Nachdem das Vorstellen der Motivation abgeschlossen ist, folgen im nächsten Kapitel die Grundlagen. Ziel des Abschnitts ist es, eine kurze Einführung in die Ansätze der künstlichen neuronalen Netze sowie der Fuzzy-Systeme zu geben. Aufbauend auf den Stärken und Problemen der erläuterten Technologien werden anschließend die eigentlichen Neuro-Fuzzy-Systeme vorgestellt. Da sich der Fokus dieser Ausarbeitung auf die hybriden Systeme richtet, werden an dieser Stelle ANFIS und NEFCON als Beispiele für die Umsetzung dieser Technologie vorgestellt. Abschließend werden die Ergebnisse kurz zusammengefasst und kritisch bewertet.

[1] Universität Oldenburg, VLBA, Ammerländer Heerstraße 114-118, 26129 Oldenburg, stefan.zurborg@uni-oldenburg.de

2 Grundlagen

Nachfolgend werden die Grundlagen erläutert, die für das Verständnis dieser Ausarbeitung notwendig sind. Dafür werden *künstliche neuronale Netze* und *Fuzzy-Systeme* vorgestellt, wobei auch deren Vor- und Nachteile untersucht werden.

2.1 Künstliche neuronale Netze

Künstliche neuronale Netze sind ein Teilbereich der KI und versuchen die Funktionalitäten menschlicher Neuronen zu imitieren, um deren Lernfähigkeit für Probleme anzuwenden, die nicht durch mathematische Modelle beschrieben werden können. Der Aufbau eines künstlichen neuronalen Netzes setzt sich in der Regel aus mehreren Ebenen zusammen. Diese bestehen aus je einer Ein- und Ausgabeschicht sowie eventuell aus verborgenen Schichten zwischen ihnen [Wa18].

Die Verbindung zwischen den einzelnen Schichten (Kanten) sind unterschiedlich gewichtet und bestimmen dadurch den Einfluss eines Signals auf die nächste Ebene. Im Verlauf des Lernprozesses (Trainingsphase) bestimmen Lernregeln die Gewichtung der Kanten in Abhängigkeit von Ein- und Ausgabesignalen und passen diese gegebenenfalls an. Dementsprechend ist das „Wissen“ eines neuronalen Netzes in diesen Gewichtungen gespeichert [Lä03].

Künstliche neuronale Netze bieten demnach die Möglichkeit, komplexe Problemstellungen ohne ein mathematisches Prozessmodell oder vorheriges Regelwissen durch verschiedene Lernalgorithmen zu bearbeiten. Neben diesen Stärken bestehen dennoch einige Hindernisse bei ihrem Einsatz [Bo03].

Das trainierte neuronale Netz stellt für den Nutzer eine Black-Box dar, sodass nach dem Lernprozess kein Regelwissen aus dem Netz extrahiert werden kann. Darüber hinaus können Anpassungen an veränderte Parameter in der Regel nur durch das Wiederholen des Lernvorgangs vorgenommen werden. Ein elementarer Unterschied zu den Fuzzy-Systemen besteht darin, dass neuronale Netze kein Vorwissen verwenden. Muster und Regeln müssen demnach immer neu erlernt werden [Bo03].

2.2 Fuzzy-Systeme

Die Fuzzy-Systeme stellen einen vielversprechenden Ansatz dar, um die Funktionsweise künstlicher neuronaler Netze zu ergänzen und dadurch entsprechende Nachteile auszugleichen. Im Folgenden werden das Grundkonzept der Fuzzy-Logik sowie exemplarische Fuzzy-Regler als Implementierungen in Fuzzy-Systemen vorgestellt.

Die klassische Logik unterteilt Wahrheitswerte in die zwei Kategorien *wahr* und *falsch*. Diese Beschreibung ist für die Lösung komplexer Probleme, welche auf sprachlichen

Angaben beruhen, hinderlich. Die Begriffe der menschlichen Sprache sind vage und dadurch selten binär festgelegt. Beispielsweise wird die Wassertemperatur von Menschen nicht nur durch „warm“ und „kalt“ beschrieben, sondern es existieren Abstufungen wie „lauwarm“. Dadurch ist auch die klassische Mengenlehre, nach der ein Element entweder innerhalb oder außerhalb einer Menge liegt, oft unangemessen [Bo03].

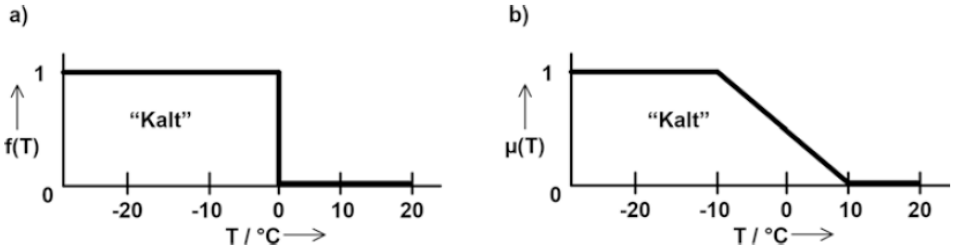


Abbildung 1: Temperaturverlauf nach (a) Boolescher Logik und nach (b) Fuzzy-Logik [SRN17a]

Die Fuzzy-Logik erweitert diese klassische Logik, damit linguistische Ausrücke mathematisch erfasst werden können. Die Funktionsweise dieser Thematik wird in Abbildung 1 anhand eines Beispiels veranschaulicht. Im ersten Teil der Abbildung ist die Beschreibung der Temperatur nach der klassischen Logik und Mengenlehre aufgeführt. Temperaturen unterhalb des Gefrierpunktes werden als „kalt“ definiert und dementsprechend dieser Menge zugeordnet. Höhere Temperaturen sind folglich nicht mehr Teil dieser Menge. Im zweiten Teil der Abbildung wird dieses Problem durch die Fuzzy-Logik beschrieben. Die extremen Temperaturen werden dabei weiterhin eindeutig einer Menge zugeordnet. Allerdings wurde der Grenzbereich beim Gefrierpunkt um eine Zugehörigkeitsfunktion erweitert, welche den Grad der Zugehörigkeit zu den Mengen angibt und somit die scharfe Trennung der klassischen Logik auflöst. Dadurch verhält sich die Fuzzy-Logik im Bereich der Grenzwerte wie die klassische Logik. Da zusätzlich auch die Zwischenwerte monoton verlaufen, bleiben viele Gesetze der klassischen Logik erhalten [SRN17a].

Die größten Erfolge von Fuzzy-Systemen im Bereich der industriellen und kommerziellen Anwendungen wurden durch Fuzzy-Regler erzielt. Diese können einen nicht-linearen Regler definieren, indem man eine nicht-lineare Übertragungsfunktion angibt, welche nicht jeden einzelnen Tabelleneintrag speichern muss. Fuzzy-Regler bestehen demnach aus einer Menge unpräziser Regeln, welche für eine wissensbasierte Interpolation eine unpräzise definierte Funktion verwenden. Die Regeln bestehen dabei grob aus einer Prämisse und einer Konklusion. Während die Prämisse die Voraussetzung einer Regel beschreibt (WENN-Teil), kennzeichnet die Konklusion deren Schlussfolgerung (DANN-Teil). In hybriden Systemen werden vor allem der Mamdani-Regler und der Takagi-Sugeno-Kang-Regler (TSK) verwendet, welche im Folgenden kurz vorgestellt werden [Kr15a].

Mamdani-Regler: Das erste Modell eines Fuzzy-Systems basiert auf einer endlichen Menge von oben genannten WENN-DANN-Regeln, die in der folgenden Form angegeben werden:

R: WENN x_1 IST $\mu^{(1)}$ UND … UND x_n IST $\mu^{(n)}$ DANN y IST μ

$x_1, \ldots, x_n$ sind dabei die Eingangsgrößen und y die Ausgangsgröße des Reglers. Die Fuzzy-Mengen $\mu^{(1)}$ und $\mu^{(n)}$ stehen für unpräzise linguistische Werte, wie beispielsweise „ungefähr null“. Das spezielle Kennzeichen dieses Reglers ist allerdings die Interpretation der Regeln. Die einzelnen Regeln werden nicht als logische Implikationen aufgefasst, sondern als eine stückweise definierte Funktion interpretiert.

Takagi-Sugeno-Kang-Regler: Der zweite relevante Fuzzy-Regler verwendet Regeln in der folgenden Form:

R: WENN x_1 IST $\mu^{(1)}$ UND … UND x_n IST $\mu^{(n)}$ DANN $y = f_1(x_1, x_2)$

Wie schon beim Mamdani-Regler werden auch hier die Eingangswerte $(x_1,\ldots, x_n)$ unscharf beschrieben und mit linguistischen Werten angegeben. Im Gegensatz zum Mamdami-Regler wird die Konklusion nicht durch eine Fuzzy-Menge, sondern durch eine von den Eingangswerten abhängige Funktion beschrieben.

Fuzzy-Systeme bieten zusammenfassend die Möglichkeit, (Regel-) Wissen für Anwender nutzbar zu machen, ohne dass ein mathematisches Prozessmodell notwendig ist. Darüber hinaus können diese Systeme mit geringem Aufwand implementiert werden und deren Ergebnisse sind interpretierbar [Bo03].

Im Gegensatz zu den künstlichen neuronalen Netzen muss allerdings Regelwissen verfügbar sein, da Fuzzy-Systeme nicht lernfähig sind. Außerdem ist die Abstimmung des Systems mit der zu bearbeitenden Aufgabe anspruchsvoll und demnach anfällig für Fehler. Die Anpassung an veränderte Parameter ist komplex und bei der Interpretation dieser veränderten Systeme können semantische Probleme auftreten [Bo03].

3 Neuro-Fuzzy-Systeme

Nachdem das Vorstellen der beiden zugrunde liegenden Kernkonzepte abgeschlossen ist, erfolgt an dieser Stelle die Einführung in die Neuro-Fuzzy-Systeme. Dafür werden zunächst die Gründe für die Zusammenführung der beiden Konzepte vorgestellt, um darauf aufbauend die hybriden Systeme sowie deren Implementierungen ANFIS und NEFCON zu erläutern.

Die Grundidee für den Einsatz von Neuro-Fuzzy-Systemen besteht in dem Bestreben, die Vorzüge der künstlichen neuronalen Netze sowie der Fuzzy-Systeme zu kombinieren, um dadurch auch die entsprechenden Nachteile der jeweiligen Konzepte auszugleichen. Die jeweiligen Vor- und Nachteile der beiden Modelle wurden bereits in Kapitel 2 aufgeführt. Das Hauptaugenmerk liegt dabei auf der Lernfähigkeit der neuronalen Netze sowie der Interpretierbarkeit der Fuzzy-Systeme [SRN17b].

3.1 Hybride Systeme

Neuro-Fuzzy-Systeme können zwischen kooperativen und hybriden Modellen unterschieden werden. Kooperative Modelle trennen die Aufgabenbereiche der beiden Komponenten strikt voneinander ab. In diesem Modell erzeugt das neuronale Netz die Regelungsparameter oder optimiert sie während der Regelung. Hybride Modelle vereinen die beiden Konzepte direkt miteinander. Ziel dieser Architektur ist es, dass die hybriden Fuzzy-Regler auch als neuronales Netz interpretiert werden können und unter Umständen auch in einem solchen implementiert werden können. Durch diese integrierte Struktur ist keine Kommunikation zwischen den beiden Komponenten erforderlich, wodurch diese Systeme theoretisch sowohl online als auch offline lernen können. Durch diese Vorzüge konnten sich hybride Systeme gegenüber den kooperativen Systemen mehr durchsetzen [Kr15b].

3.2 ANFIS

Das erste Beispiel für die Implementierung eines Neuro-Fuzzy-Systems ist ein Modell für feste Lernaufgaben. Diese Modelle versuchen, die Fuzzy-Mengen und die Parameter der Ausgabefunktionen bei TSK-Modellen unter Berücksichtigung von Ein-Ausgabe-Tupeln zu optimieren. Dementsprechend sind diese Verfahren dann sinnvoll, wenn bereits grobe Beschreibungen in Form von Regeln vorliegen und diese auf Basis von Messdaten optimiert werden sollen. Ein Anwendungsfall besteht darin, bereits bestehende Regler durch dieses Modell abzulösen und deren Einstellungen als Startwert für das lernende System zu übernehmen. Sofern keine Fuzzy-Regelbasis vorhanden ist oder nicht erstellt werden kann, müssen beispielsweise evolutionäre Algorithmen angewendet werden, um eine initiale Regelbasis zu finden. Das folgende System ist ein typisches Beispiel für ein Neuro-Fuzzy-System, welches feste Lernaufgaben bearbeitet [Kr15b].

Das Adaptive-Network-based Fuzzy Inference System (ANFIS) von JANG [JSM97] ist ein hybrides System, welches beispielsweise zur Funktionsapproximation eingesetzt wird. Es basiert auf einem Fuzzy-System vom TSK-Typ, das in einem vorwärtsbetriebenen fünfschichtigen neuronalen Netz implementiert ist (siehe Abbildung 2) [Bo03]. Die Funktionsweise des ANFIS-Systems wird im Folgenden anhand eines Beispiels erläutert, welches an JANG angelehnt ist [JSM97]:

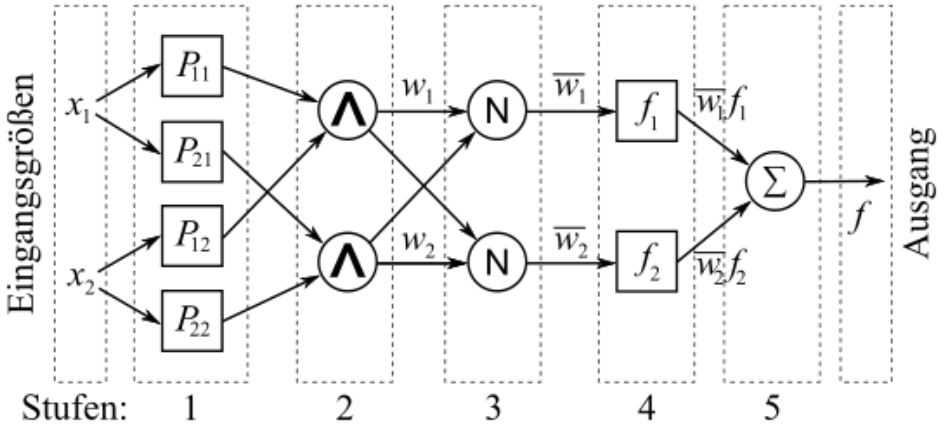


Abbildung 2: Schematische Darstellung eines ANFIS-Modells [JSM97]

In diesem einfachen Beispiel wird davon ausgegangen, dass dieses Modell aus zwei Eingangs- und einer Ausgangsgröße besteht. Darüber hinaus existieren zwei Regeln, welche nachfolgend dargestellt sind:

R_1: WENN $P_{11}(x_1)$ UND $P_{12}(x_2)$, DANN $f_1(x_1, x_2)$

R_2: WENN $P_{21}(x_1)$ UND $P_{22}(x_2)$, DANN $f_2(x_1, x_2)$

Stufe 1: x_1 und x_2 sind die Eingangsgrößen für die erste Ebene. Dabei ist zu beachten, dass weder Eingangs- noch Ausgangsgrößen zu den fünf Stufen zählen. Die Einheiten in dieser Stufe (P_{11}, P_{21} etc.) repräsentieren linguistische Terme, die mit den Regeln verbunden sind (Beschreibungen wie „klein“ oder „groß“). Der Output dieser Schicht beschreibt, zu welchem Grad der Input den Anforderungen der Regeln genügt [JSM97]. Aus mathematischer Sicht speichert jeder Knoten somit Parameter für die Definition einer Zugehörigkeitsfunktion zur Modellierung eines linguistischen Ausdrucks. Dabei ist jeder Knoten mit genau einer Eingabeeinheit verbunden und berechnet den Zugehörigkeitsgrad des aktuellen Eingabewertes [Bo03].

Stufe 2: Für jede Regel ist in der zweiten Stufe ein Knoten angelegt. In diesem wird das Produkt der eingehenden Signale gebildet und an die nächste Stufe weitergegeben. Durch das multiplizieren der Zugehörigkeitsgraden wird der Erfüllungsgrad der Regel bestimmt [Bo03].

Stufe 3: In der dritten Stufe werden die vorherigen Ergebnisse übernommen und es ist ebenfalls ein Knoten für jede Regel implementiert. In diesem wird der relative Erfüllungsgrad w ihrer Regel, bezogen auf alle anderen Regeln, berechnet. Dementsprechend ist jede Einheit dieser Stufe mit allen Einheiten der vorherigen Stufe verbunden und gibt einen normalisierten Erfüllungsgrad an die nächste Stufe weiter [Bo03].

Stufe 4: Die vierte Stufe besteht wieder aus einem Knoten pro Regel, welcher den normalisierten Erfüllungsgrad sowie die Eingangsgrößen als Input entgegennimmt[2]. Auf Basis dieser Werte wird die gewichtete Ausgabe der einzelnen Regeln berechnet [Bo03].

Stufe 5: Als finale Stufe vor der Ausgabeeinheit berechnet der letzte Knoten die Ausgabegröße als Summe aller Ausgaben der vorherigen Stufe [Bo03].

Für das Erlernen der festen Aufgabe benötigt das ANFIS-Modell eine ausreichende Menge an Ein- und Ausgabedaten. Erst durch diese Trainingsdaten können die Modellparameter der Fuzzy-Mengen sowie der Ausgabefunktion angepasst werden. Als mögliche Lernverfahren für diese Systeme kann auf bekannte Methoden der künstlichen neuronalen Netze, wie beispielsweise das Gradientenverfahren oder die Kleinste-Quadrate-Methode, zurückgegriffen werden. Zusätzlich besteht auch die Möglichkeit eine Kombination aus unterschiedlichen Lernverfahren anzuwenden [Kr15b].

Für die praktische Anwendung von ANFIS existieren allerdings einige Hindernisse, die zu beachten sind. Bezüglich der Optimierung der Fuzzy-Mengen in den Regelprämissen bestehen keinerlei Restriktionen. Dadurch können eventuell Definitionslücken auftreten, wenn der Eingabebereich nach der Optimierung nicht mehr vollständig abgedeckt wird. Als Folge ist nach der abgeschlossenen Optimierung eine Überprüfung der Ergebnisse erforderlich. Außerdem werden die Fuzzy-Mengen unabhängig voneinander optimiert, sodass die ursprüngliche Reihenfolge in der Partition verändert werden kann [Kr15b]. Darüber hinaus ist das ANFIS-Modell speziell auf die Funktionsapproximation ausgerichtet und das aufwendige Lernverfahren ermöglicht dafür auch exakte Ergebnisse. Allerdings wird dafür ein TSK-Regler verwendet, welcher die Interpretierbarkeit, ein wesentliches Merkmal der Neuro-Fuzzy-Systeme, erschwert [Bo03].

3.3 NEFCON

Das zweite Beispiel für die Implementierung eines Neuro-Fuzzy-Systems verwendet das Modell des verstärkenden Lernens. Dieses Vorgehen minimiert die Menge der Informationen, welche zum Lernen des Systems benötigt werden. Während das Lernen mit einer festen Lernaufgabe vorgegebene Stellwerte benötigt, müssen hierbei keine Parameter voreingestellt sein. Für die Anpassung des Systems genügt die Information, ob eine Lernaktion in die richtige Richtung geht. Die Implementierungen dieses Lernverfahrens basieren auf dem Prinzip, das Lernproblem in zwei Bereiche aufzuteilen. Diese bestehen zum einen aus einem kritisierenden System (Kritiker) und zum anderen aus einem System, welches die Beschreibung speichert und anwendet (Aktor). Die besondere Aufgabe des Kritikers besteht in der Bewertung der Regelzustände, um bei Bedarf Anpassungen zu erzwingen. Diese Lernmethode wird häufig in Kombination mit künstlichen neuronalen Netzen angewendet, welche im folgenden Beispiel Verwendung finden [Kr15b].

[2] Der Empfang der Eingangswerte fehlt in Abbildung 2.

Das NEFCON-Modell (NEural Fuzzy CONtroller) ist ein hybrider Ansatz für einen Neuro-Fuzzy-Regler auf der Basis eines Mamdani-Reglers. Mit diesem Modell soll während der aktiven Reglung eine interpretierbare Regelbasis durch eine möglichst kurze Trainingsphase erlernt werden. Im Gegensatz zu vielen anderen Ansätzen des verstärkenden Lernens ist auch die Integration von Vorwissen möglich, um den Trainingsaufwand zu verringern [Bo03].

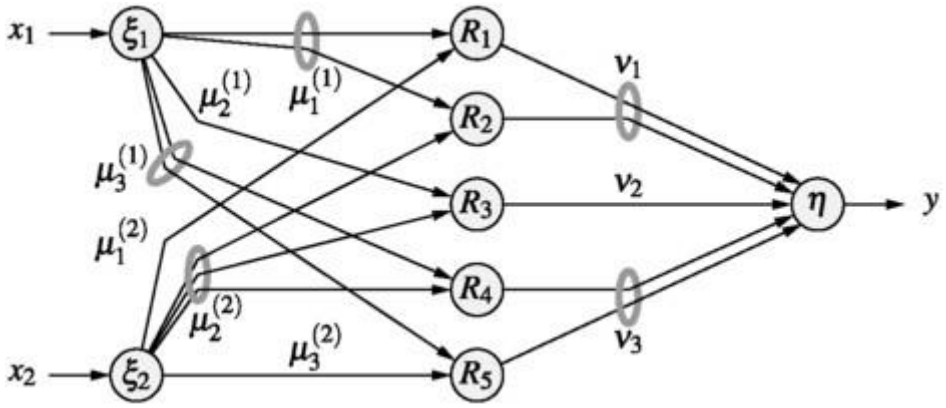


Abbildung 3: NEFCON-System mit zwei Eingangsgrößen und fünf Reglern [Kr15b]

Das NECON-Modell wird in der Regel durch eine beliebige Menge an Messgrößen, mehrere linguistische Regeln und einer Ausgabegröße beschrieben. Zusammen bildet die Netzwerkstruktur ein mehrschichtiges Perzeptron mit drei Schichten. Die Architektur dieses technischen Systems wird im Folgenden anhand des Beispiels aus Abbildung 3 beschrieben [Bo03].

Die Abbildung veranschaulicht einen Fuzzy-Regler mit zwei Eingangsgrößen (x_1, x_2), fünf Regel-Neuronen (R_1, R_2 etc.) und einer Ausgangsgröße (y). In dieser Darstellung entspricht es einem vorwärtsbetriebenen dreischichtigen neuronalen Netz [Bo03]. Die Regeln, welche die Neuronen der mittleren Schicht darstellen, sind nachfolgend aufgeführt [Kr15b]:

R_1: WENN x_1 IST $A_1^{(1)}$ UND x_2 IST $A_1^{(2)}$ DANN y IST B_1

R_2: WENN x_1 IST $A_1^{(1)}$ UND x_2 IST $A_2^{(2)}$ DANN y IST B_1

R_3: WENN x_1 IST $A_2^{(1)}$ UND x_2 IST $A_2^{(2)}$ DANN y IST B_2

R_4: WENN x_1 IST $A_3^{(1)}$ UND x_2 IST $A_2^{(2)}$ DANN y IST B_3

R_5: WENN x_1 IST $A_3^{(1)}$ UND x_2 IST $A_3^{(2)}$ DANN y IST B_3

Die Kanten μ und ν zwischen den Neuronen entsprechen den Fuzzy-Mengen aus den Prämissen bzw. den Konklusionen. Beispielsweise repräsentiert die Kante $\mu_3^{(1)}$ den Term $A_3^{(1)}$ aus der vierten und fünften Regel. An dieser Stelle wird eine Besonderheit des NEFCON-Modells deutlich. Die Verbindungen, welche durch Ringe verbunden sind, entsprechen den gleichen Fuzzy-Mengen. Dementsprechend tauchen in den Regeln dieser verbundenen Kanten immer die gleichen linguistischen Terme neben der Eingangsgröße auf. Wenn auch nur ein Gewicht im Lernprozess angepasst wird, müssen analog dazu alle Gewichte der anderen zugehörigen Verbindungen angeglichen werden. Dadurch wird verhindert, dass derselbe linguistische Term durch unterschiedliche Fuzzy-Mengen repräsentiert wird [Bo03].

Der Lernprozess des NEFCON-Modells besteht aus zwei Teilen, welche aus dem Trainieren von Fuzzy-Mengen (Parameterlernen) und dem Erlernen von Fuzzy-Regeln bestehen (Strukturlernen). Beide Phasen verwenden einen Fuzzy-Fehler, welcher die Güte der aktuellen Ergebnisse bewertet und dadurch die Rolle des Kritikers übernimmt. Falls eine Regelbasis nicht vorhanden oder unvollständig ist, erfolgt zunächst das Strukturlernen. Andernfalls kann direkt mit dem Parameterlernen begonnen werden [Bo03].

Strukturlernen: Für diesen Lernvorgang stehen unterschiedliche Verfahren zur Verfügung, deren Auswahl von der Vollständigkeit der vorhandenen Regelbasis abhängt. Ein mögliches Verfahren ist die Top-Down-Methode, welche aus zwei Teilschritten besteht. Als Grundlage für dieses Verfahren werden alle Regeln erstellt, welche aus den Eingangs- und Ausgangsgrößen gebildet werden können. Im ersten Schritt werden auf dieser Basis alle Regeln eliminiert, deren Ausgabe ein falsches Vorzeichen enthält. Aus diesem Ergebnis werden im zweiten Schritt zufällig Regeln mit gleicher Prämisse ausgewählt. Im Anschluss wird der Ausgabefehler des gesamten Systems mit der Aktivierung der individuellen Regel akkumuliert. Abschließend selektiert das System die Regel mit der kleinsten Fehlerrate und beginnt wieder mit dem ersten Teilschritt [Kr15b].

Parameterlernen: Ziel dieses Verfahrens ist das Verändern der Zugehörigkeitsfunktionen von NEFCON, um das Regelverhalten zu optimieren. Dafür wird auf die Fehler-Rückpropagation zurückgegriffen, wie sie auch bei mehrschichtigen neuronalen Netzen verwendet wird. Dieses Verfahren optimiert sowohl die Prämissen als auch die Konklusion der einzelnen Regeln. Zusammengefasst werden die Fuzzy-Mengen einer Regel, abhängig vom Beitrag zur Stellgröße und dem resultierenden Fehler, „belohnt“ oder „bestraft“. Aufgrund dieses verstärkenden Lernens passt sich die Zugehörigkeitsfunktion an die Eingangsgrößen an [Bo03].

Bei der Anwendung dieses Ansatzes sind einige Hindernisse zu beachten. Der Fuzzy-Fehler muss sehr genau definiert werden, da andernfalls Probleme bei der Bewertung auftreten können. Allerdings gestaltet sich die Festlegung des Fehlers bei einigen Aufgaben als schwierig bis unmöglich. Demnach eignet sich dieses Modell vor allem für einfache Regelstrecken oder als Basis für komplexere Probleme [Kr15b].

4 Fazit und Ausblick

Diese Seminararbeit entstand im Rahmen der Projektgruppe *Propose.AI*. In diesem Projekt sollen Technologien der künstlichen Intelligenz analysiert und bei der *Brille24 GmbH* eingeführt werden. Abschließend werden die Ergebnisse dieser Ausarbeitung in diesem Kapitel zusammengefasst und kritisch bewertet.

Neuro-Fuzzy-Systeme stellen einen interessanten Ansatz dar, um die Vorteile zweier unterschiedlichen Technologien zu vereinen. Durch die Kombination von künstlichen neuronalen Netzen und der Fuzzy-Logik entstehen lernfähige System, welche zugleich Interpretierbarkeit aufweisen. In vergangenen Jahren haben sich dabei die hybriden Systeme gegenüber kooperativen Systemen durchgesetzt. Diese bieten die Möglichkeit, unterschiedliche Lernverfahrenen, wie Modelle für feste Lernaufgaben oder für das verstärkende Lernen, zu implementieren. Dadurch können unterschiedliche Aufgaben, wie Funktionsapproximationen oder Klassifizierungen, bewältigt werden. Obwohl die Neuro-Fuzzy-Systeme einen vielversprechenden Ansatz darstellen, sind gewisse Limitationen bei deren Implementierungen zu beachten. Beispielsweise lassen sich Fuzzy-Fehler bei komplexen Problemen im NEFCON-Modell nur schwer festlegen, sodass dieses Modell, entgegen der Zielsetzung, anspruchsvoll in der Anwendung sein kann.

Wie bei allen Verfahren der künstlichen Intelligenz hängt auch bei den hybriden Systemen der Nutzen stark von der Anwendung ab. Dabei geht das Anwendungsfeld über die hier vorgestellten industriellen Regler-Systeme hinaus und so können beispielsweise ANFIS-Modell für Prognoseverfahren implementiert werden.

5 Literaturverzeichnis

[Bo03] Borgelt, C. et al.: Neuro-Fuzzy-Systeme. Von den Grundlagen künstlicher Neuronaler Netze zur Kopplung mit Fuzzy-Systemen. Vieweg+Teubner Verlag, Wiesbaden, 2003.

[JSM97] Jang, J.-S. R.; Sun, C.; Mizutani, E.: Neuro-fuzzy and soft computing. A computational approach to learning and machine intelligence. Prentice-Hall, Upper Saddle River, NJ, 1997.

[Kr15a] Kruse, R. et al.: Fuzzy-Regelsysteme. In (Kruse, R. Hrsg.): Computational Intelligence, 2015.

[Kr15b] Kruse, R. et al.: Hybride Systeme zur Optimierung von Fuzzy-Reglern. In (Kruse, R. Hrsg.): Computational Intelligence, 2015.

[Lä03] Lämmel, U.: Data Mining mittels künstlicher neuronaler Netze. Hochschule Fachbereich Wirtschaft, Wismar, 2003.

[SRN17a] Styczynski, Z. A.; Rudion, K.; Naumann, A.: Fuzzy-Logik. In (Styczynski, Z. A.; Rudion, K.; Naumann, A. Hrsg.): Einführung in Expertensysteme. Springer Berlin Heidelberg, Berlin, Heidelberg, 2017; S. 85–130.

[SRN17b] Styczynski, Z. A.; Rudion, K.; Naumann, A.: Neuro-Fuzzy-Systeme. In (Styczynski, Z. A.; Rudion, K.; Naumann, A. Hrsg.): Einführung in Expertensysteme. Springer Berlin Heidelberg, Berlin, Heidelberg, 2017.

[Wa18] Wartala, R.: Praxiseinstieg Deep Learning. Mit Python, Caffe, TensorFlow und Spark eigene Deep-Learning-Anwendungen erstellen. O'Reilly, Heidelberg, 2018.

Business Intelligence – Integration heterogener Datenquellen

Felix Jedebrock[1]

Abstract: In dieser Ausarbeitung, die im Zuge der Seminarphase der Projektgruppe PROPOSE.AI erstellt wurde, geht es um die Integration heterogener Datenquellen in das eigene Zielsystem. Dabei wird im Allgemeinen auf das Business Intelligence eingegangen, um danach tiefergehend den ETL-Prozess und Ontologien zu erklären. Bei dem ETL-Prozess werden dabei die einzelnen Komponenten (Extraktion, Transformation und Load) des Prozesses nahegebracht und zudem geschildert, welche Konflikte bei der Integration externer, heterogener Daten auftreten können. Darauf aufbauend wird näher auf die Ontologien eingegangen. Neben einer allgemeinen Erklärung folgt eine spezielle Auseinandersetzung mit den einzelnen Bestandteilen und wie mittels des Tools Protéjé eine Ontologie erstellt werden kann.

Keywords: Business Intelligence; Datenintegration; ETL-Prozess; Ontologien

1 Einleitung

Unser Jahrhundert ist geprägt von einem immer weiter ansteigenden Datenvolumen, verursacht durch die modernen Mess- und Sensordaten, aber auch die Zunahme der Daten, die aus dem Web gewonnen werden können. Zudem spielt die orts- und zeitunabhängige Kommunikation, virtuelle Arbeitsplätze sowie personalisierte Daten und Informationen eine immer größere Rolle. Hinzu kommen die verkürzten Planungs- und Entscheidungsdauern, die durch Globalisierung, Dynamik und Interaktion von Wirtschaft und Gesellschaft gefordert sind. Dies hat auch Auswirkungen auf die Informations- und Kommunikations-Planung im IT-Bereich, welche sich auf das gesteigerte Datenvolumen bzw. Big Data, aber auch die erhöhte Erfassungs-, Mess- und Veränderungsgeschwindigkeit der Daten sowie die ansteigende Verarbeitungskomplexität einstellen muss. Hinzu kommen die zunehmende Heterogenität und die stärkere Verteilung der Daten in Cloud-Systemen, aber auch die hohen Erwartungen der Endanwender an eine Informationsabdeckung in Echtzeit [ML13].

Daraus resultierend ergeben sich einige Anforderungen aus Sicht des Business Intelligence (BI). Darunter fallen die kostengünstige Speicherung der Daten, abteilungs- und werksübergreifende Datenanalyseverfahren sowie Kosten-Nutzen-günstige Cloud-Architekturen [ML13]. Darüber hinaus und im Allgemeinen ist zu sagen, dass BI Anwendungen und Technologien es Firmen ermöglichen, besser informierte Wirtschaftsentscheidungen zu

[1] Carl von Ossietzky Universität Oldenburg, Wirtschaftsinformatik / Very Large Business Applications (VLBA), Ammerländer Heerstraße 114-118, 26129 Oldenburg, Deutschland felix.jedebrock@uol.de

treffen und Wettbewerbsvorteile zu schaffen, indem Firmen interne und externe Daten sammeln und auswerten und somit Prognosen erstellen können [Be07]. Hierbei bedient sich das BI an Methoden und Prozessen zur Erhebung, Speicherung und systematischen Auswertung elektronischer Daten und verwendet dabei die drei grundlegenden Technologien: Data Warehouse, Online Analytical Processing (OLAP) und Data Mining [Maa].

Bei dem Data Warehouse handelt es sich um eine logisch zentrale, einheitliche und konsistente Datenbasis, die als Grundlage für verschiedene Auswertungssysteme dient und zur Unterstützung analytischer Aufgaben von Führungskräften benötigt wird. Dabei werden unternehmensinterne sowie -externe Daten mittels ETL-Prozess (extract transform load) aufbereitet um bei der Speicherung der Daten die Konsistenz und Einheitlichkeit des Datenbestandes zu gewährleisten [Ha14]. Um die heterogenen, externen Daten mit dem internen Datenbestand zu vereinigen, können Ontologien verwendet werden, die dem System logische Schlussfolgerungen z.B. zu Synonymen und Homonymen in Form von wesentlichen Begriffen und deren Relationen ermöglichen und somit den ETL-Prozess unterstützen [AL04].

Unter OLAP versteht sich ein Konzept, dass zur analytischen multidimensionalen Datenauswertung genutzt wird. Hierdurch sollen die Daten in intuitiver und realitätsnaher Form bereitgestellt werden, um die Unternehmensführung zu unterstützen. Im Gegensatz zu dem Data Warehouse, in dem der Schwerpunkt auf einer einheitlichen und verfügbaren Datenbasis liegt, steht bei OLAP die Datenanalyse im Vordergrund und geht somit einen Schritt weiter in Richtung Anwendung, indem verschiede Kennzahlen multidimensional ausgewertet werden können [To00]. So wird dem Anwender die Möglichkeit geboten, eine natürliche realitätsgetreue Sicht auf sein Arbeitsumfeld zu erhalten und gleicht somit die Problematik der operativen Datenmodelle aus, welche nur unzureichend große Datenmengen für Analysezwecke handhaben können [Pe05].

Das Data Mining wird verwendet, um in vorhandenen Datenbestände nach Mustern, Trends oder Zusammenhängen zu suchen. Hierzu werden Algorithmen verwendet, die sich Erkenntnissen aus den Bereichen der Informatik, Mathematik und Statistik bedienen. Es geht darum neues Wissen zu erlangen und dadurch bei der Entscheidungsfindung bestimmter Probleme zu helfen [Tu16].

Da sich der Schwerpunkt dieser Ausarbeitung auf die Integration heterogener Daten konzentriert, wird im Folgenden allerdings nur näher auf das Data Warehouse bzw. auf den ETL-Prozess und den Umgang und die Erstellung von Ontologien eingegangen.

2 Datenintegration mittels ETL-Prozess

Bei der Datenintegration geht es darum, operative und externe Daten in das Data Warehouse zu integrieren. Unter operativen Daten verstehen sich dabei Daten, die im Unternehmen

geschäftsbedingt anfallen und meistens in einer strukturieren Form vorliegen. Externe, heterogene Daten werden hingegen aus verschiedenen anderen Quellen gewonnen und sind in der Regel nur in semi-strukturierter oder unstrukturierter Form zugänglich. Dabei nimmt das Datenvolumen von externen Daten auf Grund der sozialen Netzwerke, neuer Sensortechniken und ähnlichem rasant zu, kann in den meisten Fällen bei Einbeziehung aber auch deutliche Mehrwerte schaffen. Dabei können die Rohdaten mittels Datenvolumen, Veränderungsgeschwindigkeit, Datenvielfalt und Wahrheitswert charakterisiert werden. Entscheidendster Punkt ist dabei der Wahrheitswert und die Glaubwürdigkeit der extrahierten Daten, da es sich um heterogene Daten handelt und demensprechend in sich widersprüchlich sein können [ML13].

Der ETL-Prozess ist dabei ein zentraler Bestandteil und eine der wichtigsten Komponenten im Data Warehouse. Dabei sorgt er dafür, das Daten aus Quellsystemen in das Zielsystem überführt werden. ETL steht dabei für extract, tranform, load (übersetzt: Extraktion, Transformation, Laden). Dies sind die drei wichtigen Komponenten, denn die Quelldaten werden zunächst in der Stage-Schicht extrahiert, in der Cleanse-Schicht logisch aufbereitet bzw. transformiert und dann in die Zieltabelle geladen (Abbildung 1) [Mab]. Im Folgenden wird näher auf die einzelnen Komponenten eingegangen.

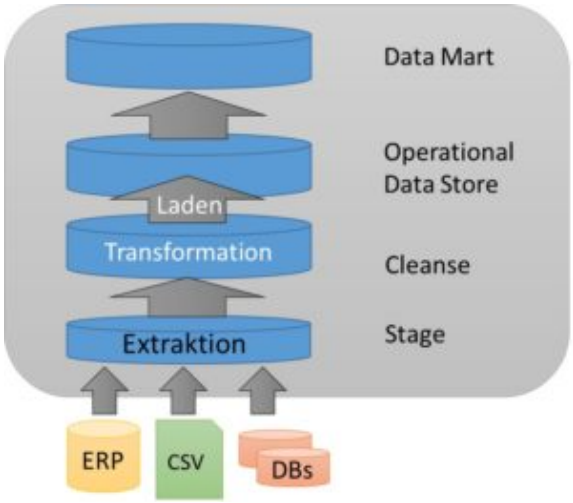


Abb. 1: Aufbau ETL-Prozess [Mab]

2.1 Extraktion

Bei der Extraktion geht es darum, dass aus unterschiedlichen Datenquellen – also relationalen Datenbanken, File-Systeme, XML-Dokumenten oder ähnlichem – Daten extrahiert und gespeichert werden [Wi12]. Dabei wird ein Zeitpunkt für die Aktualisierung der Daten

festgelegt, der zum einen zeitlich oder auch in Abhängigkeit von der Datenquelle gewählt werden kann. Aus zeitlicher Sicht können laut Farkisch folgende Methoden dazu identifiziert werden [Fa11]:

- Periodische Extraktion – benötigte Aktualität der Daten steht im Vordergrund, weswegen Datenbestand periodisch aktualisiert wird.
- Sofortige Extraktion – Anwendung erfordert aktuellsten Stand der Daten und stößt somit die Aktualisierung des Datenbestandes an.
- Anfragengesteuerte Extraktion – explizite Anfrage benötigt einen aktuellen Stand der Daten.
- Ereignisgesteuerte Extraktion – durch Auftreten eines bestimmten Ereignisses wird die Daten Aktualisierung vorgenommen.

In Abhängigkeit zu einer Datenquelle führt Farkisch weitere Punkte auf [Fark11]:

- Replizierende Quellen – synchrone Datenänderung beider Datenbestände durch den Einsatz von Technologien und Diensten zur Replikation.
- Aktive Quellen – durch die Änderung eines bestimmten Datenbestandes, kann diese die Aktualisierung des Datenbestandes mittels Triggermechanismen realisieren.
- Schnappschussquellen – Extraktion erfolgt auf eine logische Kopie des Datenbestandes und beeinträchtigt somit nicht den laufenden Betrieb, was allerdings auch einen hohen Aufwand an Speicher und Zeit für die Erstellung der Kopie bedeutet.
- Protokollierende Quelle – Datenänderungen werden aus Protokolleinträgen und Protokollinformationen entnommen, wodurch nur relevante Daten extrahiert werden.
- Exportierende Quellen – eine Kopie des Abbildes der Datenbank wird als Exportdatei dem Data Warehouse zur Verfügung gestellt.

2.2 Transformation

Bei der Transformation geht es darum, dass die erhobenen, heterogenen Daten aus den verschiedenen externen Datenquellen in ein internes Format gebracht werden, sodass sie mit dem eigenen Datenbestand kompatibel sind und Konflikte zwischen den Datenbeständen behoben werden [Mab]. Dabei können verschiedene Formen von Konflikten auftreten, die unterteilt werden können in Schemakonflikte und Datenkonflikte.

Schemakonflikte treten durch die unterschiedlichen Datenmodellierungen der einzelnen autonomen, heterogenen Datenquellen auf. Damit diese Konflikte behoben werden,

muss ein gemeinsames Schema aufgesetzt werden, welches beide Datenbestände verbindet. Das Erstellen von Ontologien kann bei diesem Punkt hilfreich sein, weswegen im Kapitel 3 näher darauf eingegangen wird. Die Schemakonflikte können zudem laut Müller und Lenz noch weiter unterteilt werden [ML13]:

- Namenskonflikte – verursacht durch die Verwendung von Synonymen und Homonymen, wodurch semantisch unterschiedliche Abbildungen realer Sachverhalte entstehen.
- Strukturelle Konflikte – hervorgerufen durch unterschiedliche Modellierungen ähnlicher Sachverhalten, resultierend aus unterschiedlichen Primärschlüsseln, unterschiedlichen oder fehlenden Objektbezeichnern sowie unterschiedlichen Attributen.
- Konflikte durch Datenrepräsentation – entstehen durch verschiedene Datentypen oder Wertebereiche, die jedoch semantisch äquivalente Attribute abbilden, oder durch Null- oder Default-Werte.

Sollte die Ursache der Konflikte bei den Daten selbst liegen, handelt es sich um Datenkonflikte, wobei auch hier laut Müller und Lenz zwischen verschiedenen Arten unterschieden werden kann [ML13]:

- Veraltete oder ungültige Daten – sollten Aktualisierungen der Daten ausbleiben, kann es dazu kommen, dass die Daten veraltet oder komplett unbrauchbar werden.
- Mehrdeutigkeit von Feldern – werden durch ungenaue Formulierungen der Entitäten erzeugt, wodurch Missverständnisse in der Deutung entstehen können.
- Fehlerhafte Daten – verursacht durch Platzhalter, nicht korrekte Erfassung oder Eintragung von Daten sowie das Fehlen von Daten

2.3 Laden

Nachdem die Daten erfolgreich transformiert und angepasst wurden, folgt nun der abschließende Schritt. Die extrahierten Daten werden in die Zieltabelle bzw. das Data Warehouse geladen. Dabei sind die beteiligten Systeme gesperrt oder nur eingeschränkt nutzbar, weshalb die Ladekomponenten höchste Performance und Effizienz aufweisen sollten, um den Ladevorgang möglichst kurz zu halten [Fa11].

3 Ontologien

Wie bereits erwähnt, kann das Erstellen von Ontologien dafür genutzt werden, die Schemata mehrerer Datenbestände zu verbinden, indem logische Verknüpfungen zwischen den

einzelnen Datensätzen gefunden und definiert werden. Dabei sind in einer Ontologie Begriffe charakterisiert und gesammelt, die von einer Gruppe von Personen für einen Anwendungsbereich als relevant erachtet werden [YMR].

Der Begriff kommt dabei aus der Philosophie und steht für die Lehre vom Sein oder genauer die Möglichkeiten und Bedingungen des Seienden und es dementsprechend eng verbunden mit der Erkenntnistheorie, die sich mit den Grenzen der menschlichen Wahrnehmung und Erkenntnis auseinandersetzt. Der Mensch ist dabei in der Lage, mit seinem gespeicherten Grund- und Kontextwissen über einen bestimmten Wissensbereich logische Schlussfolgerungen zu ziehen. Ein System hat diese Fähigkeit zunächst nicht und benötigt dementsprechend Informationen darüber, wie die Daten strukturiert und zu interpretieren sind, wozu Metadaten - in Form von Begriffen und deren Zusammenhänge - verwendet werden können, die dem System den gewünschten Kontext liefern [Wo05].

Die Ontologien haben dabei besonders durch die neue Entwicklung im Bereich des Semantic Webs an Popularität gewonnen [St11]. Dabei handelt es sich bei dem Semantic Web um eine Weiterentwicklung des World Wide Webs, bei der ein höherer Automatisierungsgrad von Geschäftsprozessen angestrebt wird. Dies bezieht sich auf standardisierte Suchanfragen, die Erbringung von zusammengesetzten Leistungen und die Verwaltung von verteilten Zugriffsrechten, welche durch eine Reihe von Schichten bzw. Ausbaustufen erreicht werden soll. Eine dieser Stufen ist das Bilden von Ontologien und Schemata, bei denen der für die Beschreibung der Strukturelement verwendete Wortschatz und die Grammatik festgelegt wird und es somit dem System erlaubt, logische Zusammenhänge zu bilden [Ro09].

3.1 Erstellen von Ontologien

Eine Ontologie besteht von der Grundstruktur aus vier Bestandenteilen: Lexikon, Begriffen, semantische Relationen und regelhaften Zusammenhängen.

Bei dem Lexikon handelt es sich um eine Menge von Worten, mit denen Begriffe und semantische Relationen bezeichnet werden. Dabei charakterisieren die Begriffe alle Begrifflichkeiten, die von einer Gruppe von Personen für einen Anwendungsbereich als relevant eingestuft werden. Durch die semantischen Relationen werden dabei die Begriffe einer Ontologie zueinander in Beziehung gesetzt. Die regelhaften Zusammenhänge können zudem einen zusätzlichen Bedeutungsinhalt von Begriffen und Relationen schaffen, die es dem System erlauben, weitere logische Schlussfolgerungen zu ziehen [YMR].

In Abbildung 2 sieht man das schemenhafte Beispiel einer Ontologie und ihrer Bestandteile. Dabei ist zu erwähnen, dass es aus Übersichtsgründen nur ein grober Ausschnitt ist und deswegen keine Vollständigkeit gegeben sein kann. Zu erkennen sind einige Begriffe,

die in Form von Klassen, Unterklassen oder Instanzen aufgeführt werden. So ist beispielsweise der Betreuer eine Unterklasse der Klasse Person und wiederum Marius Wybrands eine Instanz der Klasse Betreuer. Die verschiedenen Klassen haben dabei Eigenschaften, die an Unterklassen weitervererbt werden. Der Betreuer hat somit neben der Eigenschaft *hat Gehalt* auch die Eigenschaft *hat Geburtstag*. Um diese Beziehungen zwischen den verschiedenen Klassen und Instanzen zu Kennzeichen, werden Relationen verwendet. So wird mit der *is a*-Relation beschrieben, dass der Betreuer und der Student beide Personen sind und somit auch beide einen Geburtstag haben. Mit der *instance_of*-Relation kann gekennzeichnet werden, wenn eine Instanz von einer gewissen Klasse abstammt. Dabei kann es aber auch Relationen zwischen Instanzen geben, wie zwischen Marius und der Universität Oldenburg. In diesem Fall muss allerdings darauf geachtet werden, dass es Relationen in beide Richtungen gibt. Zum einen arbeitet Marius bei der Universität Oldenburg, zum anderen hat die Universität Marius als Mitarbeiter. Hierbei handelt es sich dementsprechend um eine rekursive Relation [Me16][NM01].

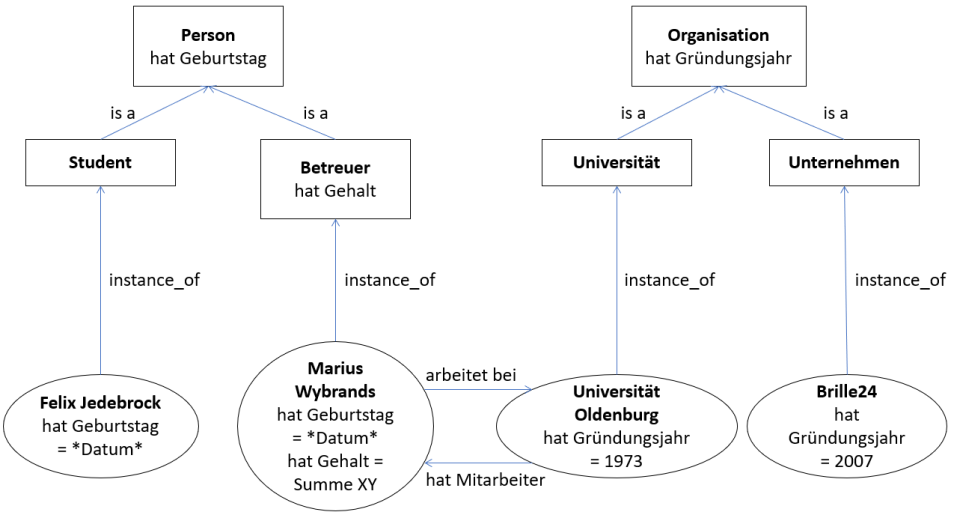


Abb. 2: Schemenhaftes Beispiel einer Ontologie (keine Vollständigkeit)

Zur Erstellung einer Ontologie können verschiedene Tools benutzt werden. Das Bekannteste ist dabei Protéjé, welches 2007 von rund 68,2 Prozent genutzt wurde [Ca07]. Es bietet dabei Werkzeuge an, die es ermöglichen, Ontologien in unterschiedlichen Formaten – wie RDF, OWL und UML – zu importieren und in das interne Wissensmodell zu überführen [Kn04].

Zunächst sollten alle benötigten Klassen unter *Classes* erstellt werden, wie in Abbildung 3 zu sehen ist. Dabei sollten die Unterklassen auch als solche gespeichert sein. Daraufhin können unter *Object properties* gewünschte Relationen eingebunden werden, wobei angegeben werden muss, welche Akteure dabei beteiligt sind. Hierzu gibt es die

Felder *Domains*, wo beschrieben wird, von dem die Aktion ausgeht, und *Ranges*, bei dem festgelegt wird, auf wen die Aktion sich auswirkt. Daraufhin können die verschiedenen Eigenschaften den Klassen unter *Data properties* angelegt werden. Wieder stehen uns dabei die beiden Felder *Domains* und *Ranges* zur Verfügung. Diesmal trägt man in ersteres die Klasse ein, für die die Eigenschaft gelten soll. Sollte es zu der gewünschten Klasse Unterklassen geben, werden diese Eigenschaften automatisch vererbt. Unter *Ranges* kann man in diesem Fall den Typen des Feldes – also String, Integer oder ähnliches – bestimmen. Abschließend hat man nun die Möglichkeit die einzelnen Instanzen unter *Individuals* einzubinden. Hierbei kann man nach dem Erstellen einer Instanz unter *Data property assertions* die einzelnen, vorher angelegten Eigenschaften der Instanz zuweisen.

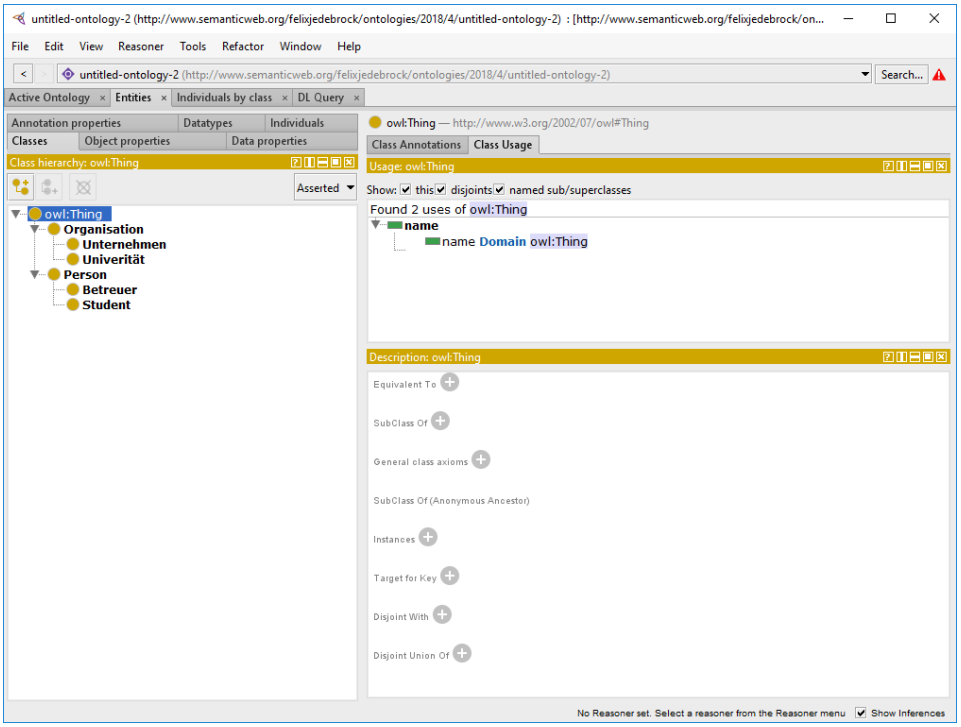


Abb. 3: Benutzeroberfläche des Tools Protéjé

4 Fazit

Die Integration heterogener Daten ist sehr umfangreich und es gibt einige Dinge, auf die enorm geachtet werden muss. Besonderes Augenmerk sollte dabei auf die Konflikte gelegt werden, die während des Transformations-Schrittes im ETL-Prozess neutralisiert werden müssen. Durch die Einhaltung des ETL-Prozesses sollte allerdings gewährleistet

sein, dass keine Problematik mit der Integration von heterogenen Daten auftreten. Dabei ist anzumerken, dass der Zeitaufwand dafür sehr hoch ist. Allerdings können auch viele Mehrwerte durch die externen Daten geschaffen werden, weswegen man den Aufwand in jedem Fall betreiben sollte.

Besonders Ontologien eignen sich hervorragend, um Struktur in die einzelnen Datenbestände zu bekommen und somit den ETL-Prozess zu unterstützen. Zudem schaffen sie einen guten Überblick über die Daten und sorgen dafür, manche Zusammenhänge durch die grafische Umsetzung besser zu verstehen.

Das Tool Protéjé ist zudem ideal für die Erstellung von Ontologien geeignet. Die Einarbeitung fällt dabei nicht schwer und die Handhabung ist nach kurzer Zeit simpel zu verstehen. Zudem gibt es Plugins die eine zusätzliche grafische Visualisierung ermöglichen.

Literaturverzeichnis

[AL04] Andreas Abecker; Ludger van Elst: Ontologies for Knowledge Management. Staab und Studer, 2004.

[Be07] Beranek, Stefan: Data Mining für Business Intelligence. 2007.

[Ca07] Cardoso, Jorge: The Semantic Web Vision: Where Are We? IEEE Intelligent Systems, 22(5):84–88, September 2007.

[Fa11] Farkisch, Kiumars: Data-Warehouse-Systeme kompakt: Aufbau, Architektur, Grundfunktionen. Xpert.press. Springer, Berlin, 2011. OCLC: 844906596.

[Ha14] Hahne, Michael: Modellierung von Business-Intelligence-Systemen: Leitfaden für erfolgreiche Projekte auf Basis flexibler Data-Warehouse-Architekturen. Edition TDWI. dpunkt-Verl, Heidelberg, 1. aufl. Auflage, 2014. OCLC: 882163134.

[Kn04] Knublauch, Holger; Fergerson, Ray W.; Noy, Natalya F.; Musen, Mark A.: The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. In (Hutchison, David; Kanade, Takeo; Kittler, Josef; Kleinberg, Jon M.; Mattern, Friedemann; Mitchell, John C.; Naor, Moni; Nierstrasz, Oscar; Pandu Rangan, C.; Steffen, Bernhard; Sudan, Madhu; Terzopoulos, Demetri; Tygar, Dough; Vardi, Moshe Y.; Weikum, Gerhard; McIlraith, Sheila A.; Plexousakis, Dimitris; van Harmelen, Frank, Hrsg.): The Semantic Web – ISWC 2004, Jgg. 3298, S. 229–243. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[Maa] Markus Begerow: , Business Intelligence - Was ist Business Intelligence? | Business Intelligence Grundlagen. URL http://www.datenbanken-verstehen.de/business-intelligence/business-intelligence-grundlagen/business-intelligence/. - abgerufen am 2018-05-06.

[Mab] Markus Begerow: , ETL (Extraktion, Transformation, Laden) im Überblick | DWH-Komponenten. URL http://www.datenbanken-verstehen.de/data-warehouse/data-warehouse-grundlagen/data-warehouse-komponenten/etl-prozess/. - abgerufen am 2018-05-07.

[Me16] Melanie Siegel: , Ontologie - Semnatik II, Juni 2016.

[ML13] Müller, Roland M.; Lenz, Hans-Joachim: Business Intelligence. eXamen.press. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[NM01] Noy, Natalya F; McGuinness, Deborah L: Ontology Development 101: A Guide to Creating Your First Ontology. S. 25, 2001.

[Pe05] Petersohn, Helge: Data Mining: Verfahren, Prozesse, Anwendungsarchitektur. Oldenbourg Verlag, 2005.

[Ro09] Rolf Grütter: Semantic Web zur Unterstützung von Wissensgemeinschaften. Januar 2009.

[St11] Stuckenschmidt, Heiner: Ontologien: Konzepte, Technologien und Anwendungen. Informatik im Fokus. Springer, Berlin, 2. aufl. Auflage, 2011. OCLC: 844787718.

[To00] Totok, Andreas: Modellierung von OLAP- und Data-Warehouse-Systemen. Dr. Andreas Totok, März 2000.

[Tu16] Tutanch: , Was ist Data Mining?, September 2016. URL https://www.bigdata-insider.de/was-ist-data-mining-a-593421/. - abgerufen am 2018-05-06.

[Wi12] Wilhelm Hummeltenberg: , ETL — Enzyklopaedie der Wirtschaftsinformatik, November 2012. URL http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/daten-wissen/Business-Intelligence/ETL. - abgerufen am 2018-05-07.

[Wo05] Wolfgang Hesse: , Ontologie(n), Juli 2005.

[YMR] York Sure; Marc Ehrig; Rudi Studer: , Automatische Wissensintegration mit Ontologien.

Betreiben und Warten von Big Data AI-Projekten

Jannik Otten

Abstract: Diese Arbeit soll aufzeigen, wie AI-Systeme auf Basis von Big Data betrieben und gewartet werden können.Hierzu werden verschiedene Architekturansätze miteinander verglichen. Anschließend werden Methoden für die Weiterentwicklung, sowie für die Integration und Bereitstellung von Änderungen vorgestellt.

Keywords: Artificial Intelligence; Big Data; CRISP-DM; Continuous Integration

1 Einleitung

Artificial Intelligence (AI) entwickelt sich in den vergangenen Jahren stark weiterentwickelt, dabei reicht die Geschichte von AI als eigene Wissenschaft bis in die Mitte des zwanzigsten Jahrhunderts zurück. Moderne AI-Systeme sind in der Lage Fahrzeuge autonom zu steuern oder eine zuverlässige Klassifizierung von Fotos vorzunehmen. (vgl. [EE16])

Elaine Rich definierte Artificial Intelligence 1983 in [Ri83] wie folgt:

> „Artificial Intelligence is the study of how to make computers do things at which, at the moment, people are better.“ [Ri83]

Hierbei bildet AI menschliche Fähigkeiten wie autonomes Entscheiden, erkennen von Mustern oder lernen ab. In manchen Bereichen ist AI bereits Menschen überlegen, in anderen ist der Mensch weit besser. (vgl. [EE16])

Ein Teilbereich von AI bildet maschinelles Lernen (ML). ML ermöglicht es Muster und Gesetzmäßigkeiten in Datenbeständen zu erkennen und basierend darauf Lösungen für vorgegebene Probleme zu entwickeln. Um ein ML-System in die Lage zu versetzen Probleme lösen zu können muss dieses zunächst mithilfe vorhandener Daten trainiert werden, im Ergebnis einstehen Modelle. Diese Modelle werden anschließend im ML-System eingesetzt, um neue Probleme zu Lösen. Die Entwicklung von Big Data-Technologien spielt hierbei eine große Rolle. Big Data ermöglicht es dem ML-System große Datenmengen schnell und effizient zur Verfügung zu stellen. (vgl. [Li16])

Diese Arbeit soll aufzeigen, wie AI-Systeme auf Basis von Big Data betrieben und gewartet werden können.

2 Betrieb

In diesem Kapitel werden im ersten Schritt verschiedene Architekturen für AI-Systeme miteinander verglichen. Hierbei wird der monolithische Ansatz einer serviceorientierten Architektur gegenübergestellt.

2.1 Architektur

Der Betrieb einer Anwendung setzt zunächst eine geeignete Architektur voraus. Folgend werden beispielhaft zwei mögliche Architekturen beschrieben.

Monolithisch vs. Serviceorientiert Eine monolithische Architektur verbindet alle Funktionen in einem untrennbaren, homogenen Gebilde. Monolithische Systeme sind häufig sehr stark an Ressourcen wie beispielsweise Datenformate gebunden. Grade bei Projekten mit kleinem Umfang eignet sich diese Architektur sehr gut, da durch die gemeinsame Codebasis einfache getestet werden kann und die Entwicklung insgesamt schneller vorangeht. Monolithen sind allerdings unflexibler als andere Architekturen und grade bei größeren Entwicklungsprojekten verlängert sich die Zeit zwischen einzelnen Softwarereleases. Abbildung 1 stellt diese Architektur beispielhaft dar. Der Client stellt seine Anfrage an das Web-Frontend, welches fest mit der Anwendungslogik des Backends verbunden ist. Die Anwendung kommuniziert wiederum mit einer nachgelagerten Datenbank. Diese Architektur setzt voraus, dass die gesamte Anwendung eine gemeinsame Sprache nutzt. (vgl. [Pa17], [Li17])

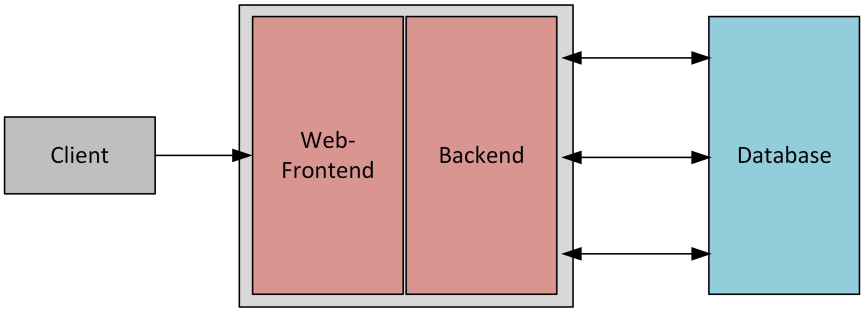


Abb. 1: Monolithische Architektur (Quelle: eigene Darstellung)

Eine mögliche Alternative zu monolithischen Systemen ist eine serviceorientierte Architektur. Dies eignet sich besonders für große Projekte mit vielen unterschiedlichen Aufgaben. Anders als bei einer bei monolithisch Anwendung werden bei einer serviceorientierten Architektur einzelne Komponenten voneinander getrennt und als Microservices betrieben. Anstatt einer großen Codebasis für die gesamte Anwendung werden die einzelnen Microservices getrennt voneinander entwickelt. Diese Herangehensweise ermöglicht es die einzelnen Komponenten

unabhängig voneinander zu skalieren und bereitzustellen. Nachteilig wirkt sich allerdings aus, dass hierdurch die Komplexität des Projektes erhöht wird. Abbildung 2 zeigt eine beispielhafte serviceorientierte Architektur. (vgl. [Pa17])

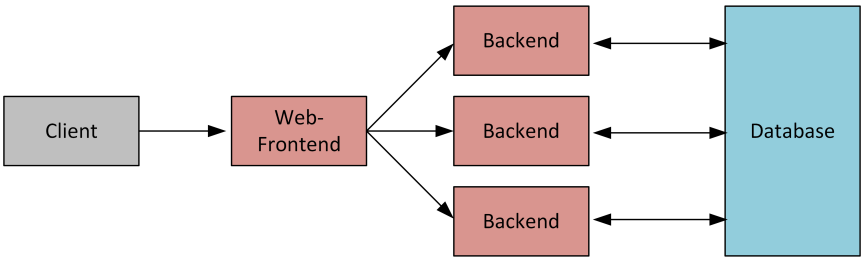


Abb. 2: Serviceorientierte Architektur (Quelle: eigene Darstellung)

Abbildung 2 zeigt beispielhaft eine serviceorientierte Architektur auf Basis von Microservices. Hierbei kommuniziert ein Client (z.B. ein Kunde eines Webshops) mit einem Webfrontend. Diese Kommuniziert mit mehreren Backends, welche verschiedene Aufgaben erfüllen. Die Backends sind an eine gemeinsame Datenbank (z.B. Hadoop als Big Data-System) angeschlossen.

Microservices kommunizieren lediglich über Schnittstellen miteinander, was es ermöglicht für verschiedene Services unterschiedliche Programmiersprachen einzusetzen (z.B. JavaScript für Webanwendungen und Python für AI/ML-Komponenten). Außerdem ermöglicht diese Architektur das gesamte Frontend oder Backend auszutauschen ohne den jeweils anderen Teil bearbeiten zu müssen. Voraussetzung hierbei ist allerdings, dass die Übermittelten Anfrage- und Antwortformate durch dem Austausch nicht verändert werden. (vgl. [Pa17])

Um Funktionen wie Logging, Authentifizierung oder Loadbalancing zu zentralisieren kann zusätzlich ein API-Gateway eingesetzt werden. Das Gateway befindet sich in der Architektur vor den Backends und verarbeitet alle API-Anfragen der vorgelagerten Komponenten (z.B. Webfrontends). Mit einem API-Gateway ist es beispielsweise möglich bestimmte APIs nur für einen eingeschränkten Nutzerkreis zur Verfügung zu stellen. Die erweiterte Architektur wird in Abbildung 3 dargestellt.

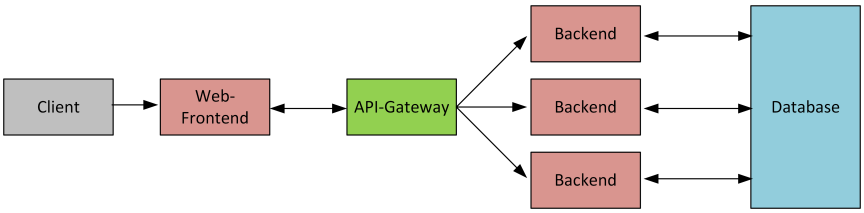


Abb. 3: Serviceorientierte Architektur mit API-Gateway (Quelle: eigene Darstellung)

Diese Architektur ermöglicht außerdem den Austausch einzelner Backendservices (z.B. bei Updates) ohne den Service einstellen zu müssen. Hierbei werden neue Anfrage an den neuen Service weitergeleitet und laufende Anfrage auf dem alten Service verarbeitet, bis

dieser die letzten Anfrage verarbeitet hat und abgeschaltet werden kann. Auf diese Weise wird die Entwicklung und Wartung der Services vereinfacht. (vgl. [Pa17])

2.2 Best Practice

Ein mögliches Anwendungsszenario ist das Erzeugen von Vorhersagen. Der Fahrdienstleister Uber nutzt beispielsweise Machine Learning, um für seinen Lieferservice UberEATS vorherzusagen wie lange eine Mahlzeit von der Zubereitung bis zur Auslieferung benötigt. Hierbei wird für jede Stufe das Prozesses (z.B. Vorbereitung der Bestellung) eine geschätzte Zeit angezeigt. Für eine möglichst präzise Vorhersage werden Bestelldaten (z.B. Tageszeit, Lieferort), historische Daten (z.B. durchschnittliche Vorbereitungszeit in den letzten sieben Tagen) sowie Echtzeitdaten (z.B. durchschnittliche Vorbereitungszeit in der letzten Stunde) verwendet. Uber setzt für diese Vorhersagen die Eigenentwicklung Michelangelo ein. Abbildung 4 zeigt das Michelangelo-System von Uber. (vgl. [HB17])

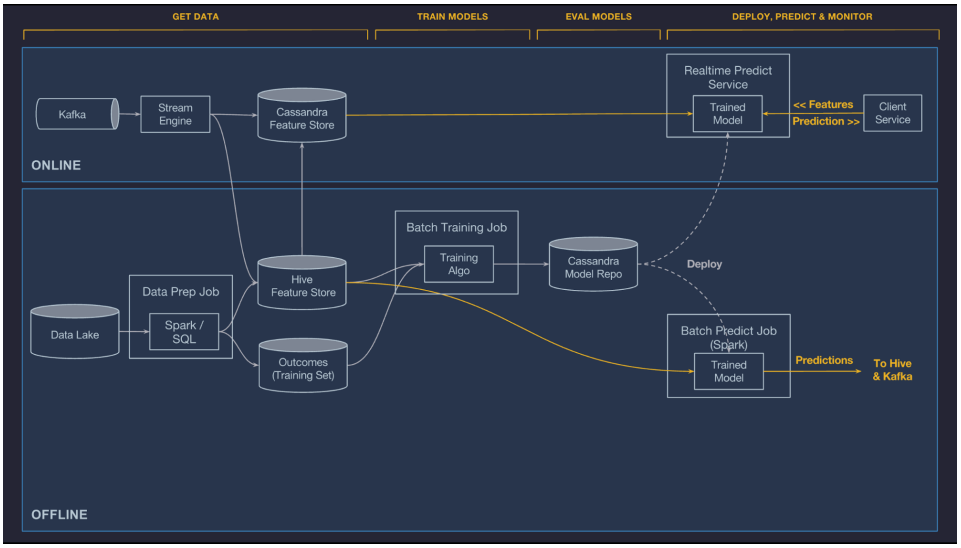


Abb. 4: Uber Michelangelo ML-Architektur (Quelle: [HB17])

Michelangelo besteht aus zwei Bereichen: Offline und Online. Im Bereich Offline werden historische Daten verarbeitet und gespeichert. Außerdem findet hier das Modell-Training statt. Das Trainierte Modell wird ein einem Repository gespeichert und in den Online-Bereich überführt. Für die Vorhersage benötigte Metriken werden in beiden Bereich in einem Feature-Store gespeichert. Im Online-Bereich werden die eingehenden Echtzeitdaten verarbeitet. Diese werden zur Speicherung in den Offline-Bereich übertragen. Anschließend erfolgt mithilfe das trainierten Modells die vorhersage der wahrscheinlichen Restzeit für den aktuellen Schritt des Lieferprozesses. (vgl. [HB17])

Mit der Michelangelo-Architektur ist Uber in der Lage eine Vorsage, bei der keine Metriken aus einem Feature-Store geladen werden müssen in weniger als 5 Millisekunden durchzuführen. Wird eine Metrik benötigt verlängert sich die Zeit auf 10ms. (vgl. [HB17])

3 Wartung von AI-Projekten

Das folgende Kapitel stellt im ersten Schritt Prozesse zur Weiterentwicklung von AI-Projekten dar. Anschließend wird beschrieben, wie Änderungen Integriert und Bereitgestellt werden können.

3.1 Prozesse zur Weiterentwicklung

Im Kontext des Knowledge Discovery in Databases (KDD) existieren bereits Prozesse, mit deinen eine kontinuierliche Verbesserung eingesetzter Modelle durchgeführt werden kann. Zu diesen zählen SEMMA (Sample, Explore, Modify, Model, Assess) und CRISP-DM (CRoss-Industry Standard Process for Data Mining) Im Folgenden wird das Modell CRISP-DM vorgestellt und auf seine Anwendbarkeit im AI-Kontext überprüft. (vgl. [AS18])

Der CRISP-DM Prozess beschriebt ein industrielles Prozessmodell, welches vor Allem im Data Mining zum Einsatz kommt.CRISP-DM besteht aus einem sechsstufigen Phasenmodell. Abbildung 5 stellt das CRISP-DM Phasenmodell dar.

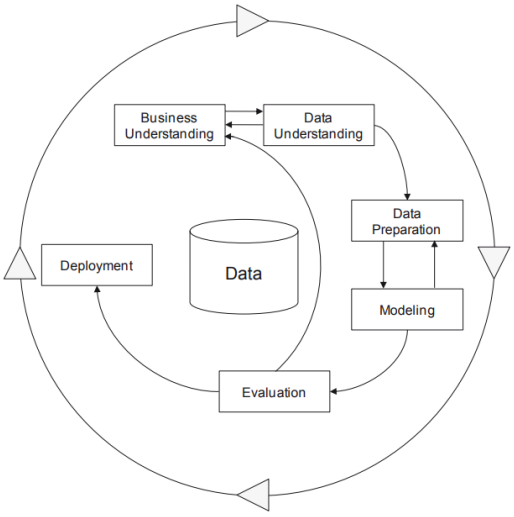


Abb. 5: CRISP-DM Phasen (Quelle: [Sh13])

1. Business Understanding Die Zielsetzung der ersten Phase ist es Anforderungen und Ziele aus betriebswirtschaftlicher Perspektive zu verstehen. Aus den betriebswirtschaftlichen Anforderungen werden Problemstellungen und Ziele, sowie Lösungswege für das Data Mining-Projekt entwickelt. (vgl. [Sh13, S.66]))

2. Data Understanding In der zweiten Phase werden erste Daten gesammelt. Diese dienen dazu einen Eindruck über die Datenqualität zu gewinnen und Hypothesen über mögliche versteckte Informationen zu bilden. (vgl. [Sh13, S.66]))

3. Data Preparation Für die Analyse ausgewählte Daten werden für die Verarbeitung zusammengestellt. Hierbei werden die Daten bereinigt und nötigenfalls transformiert. Ziel ist es die Daten in ein für die Data Mining-Software geeignetes Ausgangsformat zu bringen. (vgl. [Sh13, S.66]))

4. Modelling Anhand der Problemstellung werden Verfahren und Algorithmen ausgewählt und deren Parameter bestimmt. Am Ende des Schrittes steht eine Modell zur Datenanalyse. (vgl. [Sh13, S.66f]))

5. Evaluation Im vorletzten Schritt wird geprüft, ob die Ergebnisse des Modells qualitativ für die spezifischen Probleme geeignet ist. (vgl. [Sh13, S.67]))

6. Deployment Das gewonnene Wissen wird im Unternehmen kommuniziert. Je nach Anforderungen kann sich die Deployment-Phase verändern. Ihre Komplexität kann verschiedene Formen annehmen und bewegt sich zwischen der Erzeugung einfacher Berichte und der Implementierung ganzer Data Mining-Systeme. (vgl. [Sh13, S.67]))

Das CRISP-DM Phasenmodell kann zyklischen angewandt werden, um Data Mining Prozesse kontinuierlich zu verbessern, hierbei ist es nicht notwendig in jeden Zyklus jede Phase zu durchlaufen, so kann beispielsweise das Business Understanding übersprungen werden. Für die Verbesserung von AI-Modellen hat IBM dem CRISP-DM Lebenszyklus zusätzliche Phase hinzugefügt.

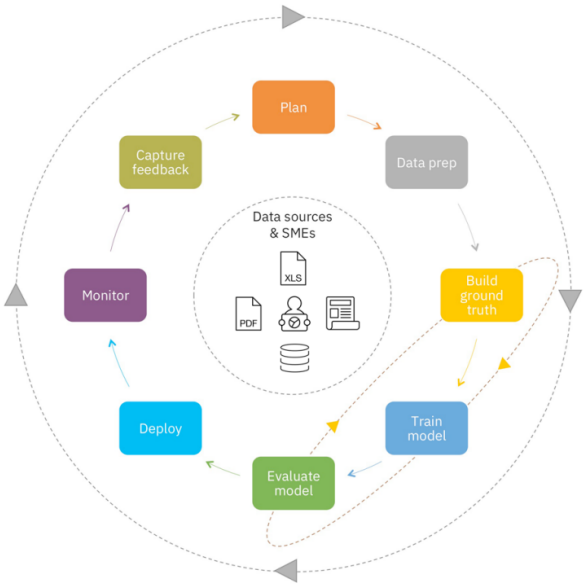


Abb. 6: CRISP-DM für AI-Projekte (Quelle: [AFC17])

Dieses in Abbildung 6 dargestellte Modell beinhaltet zusätzlich einen eigenen Zyklus für Entwicklung und Training des AI-Modells, außerdem wurde eine Überwachungsphase hinzugefügt. Durch die Zusätzlichen Schritte kann das Modell für die kontinuierliche Verbesserung von AI-Modellen verwendet werden.

3.2 Integration und Bereitstellung

In vielen Softwareprojekten wird Continuous Integration (CI) zur Weiterentwicklung verwendet. Hierbei handelt es sich um ein Konzept bei dem Entwickler mehrmals täglich Quellcode, welcher automatisiert getestet wird, in eine gemeinsame Codebasis integrieren, um so das Projekt weiterentwickeln (vgl. [Tr17]). Die Herausforderungen bei AI-Projekten stellt gegenüber normalen Softwareprojekten das Training der Modelle dar. Im Folgenden werden zwei Erweiterungen dieses Konzeptes verglichen.

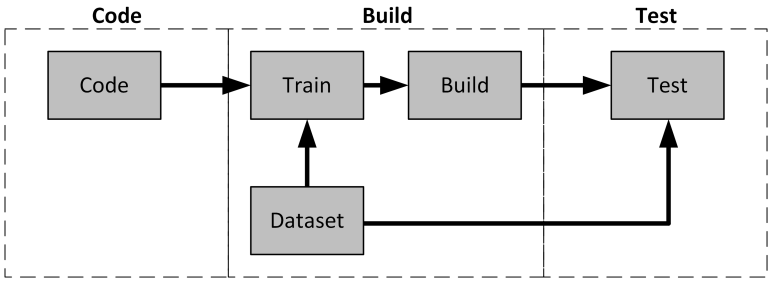


Abb. 7: Integrationsprozess für AI-Prozesse (Quelle: eigene Darstellung nach [Tv17])

Abbildung 7 zeigt einen beispielhaften Integrationsprozess für AI-Projekte, diese wurden durch einen weiteren Prozessschritt für das Training der Modelle mit einem vorgegebenen Dataset (enthält Trainings- und Testdatasets) erweitert. Am Ende erfolgt ein gemeinsamer Test der gesamten Anwendung. Diese Erweiterung lässt sich sehr gut in einen bereits vorhandenen CI-Prozess integrieren, bringt allerdings den Nachteil mit sich, dass bei jedem Zyklus das Modell erneut trainiert wird. Besonders bei Projekten in denen das Modell nicht in jedem Zyklus verändert wird verlangsamt diese Vorgehensweise den Prozess. (vgl. [Tv17])

Das beschriebene Problem wird in Abbildung 8 durch die Einführung eines getrennten Prozesses für das Modell behoben.

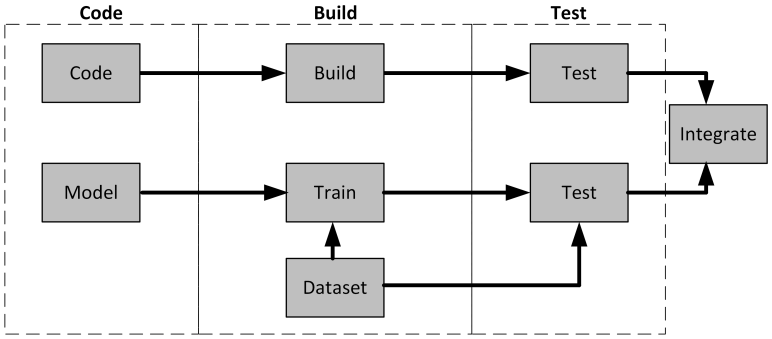


Abb. 8: Integrationsprozess für AI-Prozesse (Quelle: eigene Darstellung nach [Tv17])

Durch die Einführung von zwei getrennten Prozessen (Anwendung und Modell) muss das Training des Modells nur dann durchgeführt werden, wenn dieses angepasst wurde, außerdem werden auch die Tests der Anwendung von des Tests des Modells getrennt, was zu einer weiteren Beschleunigung des gesamten Prozesses führt. Außerdem ist es unproblematisch, wenn für Anwendung und Modell unterschiedliche Sprachen verwendet werden (z.B. Modell in Python, Anwendung in Javascript). Werden beide Prozesse am Ende mit erfolgreichen Tests abgeschlossen erfolgt die Integration in die gemeinsame Codebasis oder ein getrenntes Modell-Repository. (vgl. [Tv17])

Nachdem eine Änderung erfolgreich in die Codebasis integriert wurde erfolgt im nächsten Schritt die Überführung in den Echtbetrieb. Das AI-Frameworks TensorFlow „TensorFlow Serving “an. Hierbei handelt es sich um eine Software die es ermöglicht Modelle innerhalb von TensorFlow schnell und einfach auszutauschen. Hierzu scannt Serving periodisch das Dateisystem und lädt bzw. entlädt Modelle anhand von vorher festgelegt Regeln. Auf diese Weite ist es möglich Modelle im laufenden Betrieb auszutauschen, indem ein neues Modell kopiert wird, während das aktuelle Modell weiterhin betrieben wird. Im letzten Schritt erfolgt dann die Umschaltung auf das neue Modell. (vgl. [Ya17])

Abbildung 9 zeigt die für TensorFlow Serving benötigte Architektur.

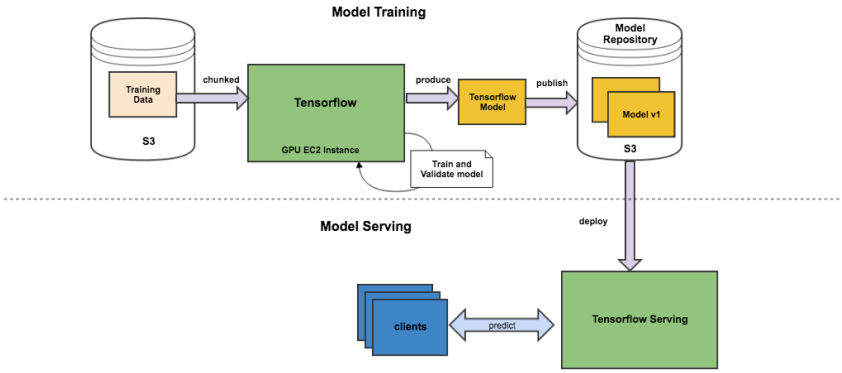


Abb. 9: Deployment am Beispiel von TensorFlow (Quelle: [Ya17])

Der obere Bereich von Abbildung 9 zeigt die Trainingsarchitektur, bestehend aus einer TensorFlow Instanz und einem Modell-Repository. Das Modell wird in der Trainingsinstanz trainiert und im Modell-Repository gespeichert. Von dort wird das Modell in die TensorFlow Serving Instanz kopiert und anschließend aktiviert. Der Zeitpunkt an dem das Training stattfindet ist für das Produktivsystem nicht relevant, die Überführung sollte allerdings dann erfolgen, wenn die Last auf dem Produktivsystem am geringsten ist. (vgl. [Ya17])

Eine weitere Möglichkeit wäre die Bereitstellung der AI-Services über Containertechnologien wie Docker. Das trainierte Modell wird hierbei in einen Container geladen und diese von Docker aufgebaut. Anschließend müsste der alte Container zu einem definierten Zeitpunkt beendet und der neue Container gestartet werden.

3.3 Versionskontrolle

Versionskontrolle spielt in Projekten mit mehreren Entwicklern eine große Rolle. Bei der Entwicklung eines AI-Projektes gibt es allerdings besondere Anforderungen an. Die Verwendung von Git ist für den reinen Quellcode problemlos möglich. Die Speicherung von Trainingsdaten für Modelle erzeugt allerdings einen großen Overhead im Git-Repository.

Eine Möglichkeit ist die Speicherung der Trainingsdaten auf einem externen Speicher. Hierbei wird das Repository entlastet, allerdings erfolgt für die Trainingsdaten keine Versionskontrolle.

Um die Verbindung zwischen Git und Speicher der Trainingsmodelle herzustellen kann eine Data Version Control (DVC) eingesetzt werden. Diese stellt die Daten in Abhängigkeit zum Quellcode und stellt zusätzlich eine Versionskontrolle für die Daten bereit. Diese wird zusätzlich zur Quellcode-Versionkontrolle (z.B. Git) eingesetzt und ermöglicht es Quellcode und Daten mit mehreren Entwicklern zu bearbeiten. (vgl. [Sh17])

Literatur

[AFC17] Allen, J.; Freed, A. R.; Chandrasekaran, S.: Adapt DevOps to cognitive and artificial intelligence systems - Apply DevOps to the training process, 2017, URL: https://www.ibm.com/developerworks/library/cc-devops-artificial-intelligence-cognitive/cc-devops-artificial-intelligence-cognitive-pdf.pdf, Stand: 27. 04. 2018.

[AS18] Azevedo, A.; Santos, M.: KDD, SEMMA AND CRISP-DM: A PARALLEL OVERVIEW, 2018, URL: http://recipp.ipp.pt/bitstream/10400.22/136/3/KDD-CRISP-SEMMA.pdf, Stand: 24. 04. 2018.

[EE16] Ertel, W. V.; Ertel, W.: Grundkurs Künstliche Intelligenz : eine praxisorientierte Einführung. Springer Vieweg, Wiesbaden, 2016.

[HB17] Hermann, J.; Balso, M. D.: Meet Michelangelo: Uber's Machine Learning Platform, 2017, URL: https://eng.uber.com/michelangelo/, Stand: 04. 05. 2018.

[Li16] Litzel, N.: Was ist Machine Learning?, 2016, URL: https://www.bigdata-insider.de/was-ist-machine-learning-a-592092/, Stand: 30. 04. 2018.

[Li17] Lipinski, K.: Monolithische Software-Architektur, 2017, URL: https://www.itwissen.info/Monolithische-Software-Architektur.html, Stand: 02. 05. 2018.

[Pa17] Palladino, M.: Microservices and API Gateways, Part 1: Why an API Gateway?, 2017, URL: https://www.nginx.com/blog/microservices-api-gateways-part-1-why-an-api-gateway/, Stand: 30. 04. 2018.

[Ri83] Rich, E.: Artificial Intelligence (McGraw-Hill series in artificial intelligence). McGraw-Hill Inc.,US, 1983.

[Sh13] Sharafi, A.: Knowledge Discovery in Databases : eine Analyse des Änderungsmanagements in der Produktentwicklung. Springer Fachmedien Wiesbaden, Wiesbaden, 2013.

[Sh17] Shridhar, K.: How to Version Control your Machine Learning task, 2017, URL: https://towardsdatascience.com/how-to-version-control-your-machine-learning-task-cad74dce44c4, Stand: 04. 05. 2018.

[Tr17] Tran, D.: Continuous Integration for Data Science, 2017, URL: http://engineering.pivotal.io/post/continuous-integration-for-data-science/, Stand: 29. 04. 2018.

[Tv17] Tvaroska, B.: DevOps Pipeline for a Machine Learning Project - Applying machine learning to DevOps, 2017, URL: https://blog.statsbot.co/machine-learning-devops-611210393c1a, Stand: 29. 04. 2018.

[Ya17] Yau, W. C.: How Zendesk Serves TensorFlow Models in Production, 2017, URL: https://medium.com/zendesk-engineering/how-zendesk-serves-tensorflow-models-in-production-751ee22f0f4b, Stand: 29. 04. 2018.

Aktuelle Anwendungsfälle Artificial Intelligence

Michael Riedel[1]

Abstract: Der nachfolgende Beitrag befasst sich mit aktuellen Anwendungen im Bereich Artificial Intelligence (nachfolgend künstliche Intelligenz oder kurz KI genannt). Ein Hauptaugenmerk dieses Artikels liegt auf Anwendungen im Bereich des Natural Language Processing (NLP). So werden basierend auf diesem Themenfeld Chatbots, Textanalysetools und das wissensbasierte System IBM Watson vorgestellt. Des Weiteren wird das autonome Fahren und Alpha Go von Google DeepMind vorgestellt sowie kurz auf Anwendungsfälle im Bereich Bilderkennung, Robotik und Militär eingegangen. Schlussendlich wird sich mit den gesellschaftlichen Auswirkungen des technischen Fortschritts auseinandergesetzt.

Keywords: Artificial Intelligence, künstliche Intelligenz, AI, KI, Chatbots, Voicebots, Autonomes Fahren, Natural Language Processing, IBM Watson, Google DeepMind

1 Einführung

Um zu verstehen was sich hinter dem Begriff der künstlichen Intelligenz (KI) verbirgt und welche Anwendungsfälle man hierunter einordnen kann, ist es erforderlich eine allgemeingültige Definition zu finden. Diesbezüglich zitiert Ertel [Er16] Elaine Rich: „Artificial Intelligence is the study of how to make computers do things at which, at the moment, people are better." [Ri88]. Nach dieser allgemeingültigen Definition wird die künstliche Intelligenz als Forschung betrachtet, die versucht Computer zu befähigen Aufgaben zu bewältigen, in denen Menschen aktuell noch besser sind. So ist es zu erklären, dass Anwendungen, die vor vielen Jahren noch als intelligent galten, heute bereits als selbstverständlich angesehen und nicht mehr der KI zugeordnet werden. Beispiele sind hier z. B. Texterkennung (OCR), Schachcomputer oder Funktionen wie Auto-Korrektur und Auto-Vervollständigung von Texten. Aus der Definition von Rich darf jedoch nicht der Schluss gezogen werden, dass sich die KI nur mit der praktischen pragmatischen Implementierung intelligenter Verfahren beschäftigt. Da die Adaptivität, die Verhaltensanpassung durch Lernen, eine besondere Stärke der menschlichen Intelligenz darstellt und der Mensch in Bezug auf die Lernfähigkeit den Computern noch weit überlegen ist, ist vor allem das maschinelle Lernen ein zentrales Teilgebiet der KI [Er16]. In dieser Ausarbeitung wird ein Überblick über aktuelle Anwendungsfälle im Bereich der künstlichen Intelligenz gegeben beginnend mit Natural Language Processing. Anschließend werden weitere Anwendungsfälle vorgestellt und abschließend werden gesellschaftliche Auswirkungen aufgezeigt.

[1] Universität Oldenburg, Fakultät 2, Ammerländer Heerstraße 114-118, 26129 Oldenburg, michael.riedel@uni-oldenburg.de

2 Aktuelle KI-Anwendungsfälle

2.1 Wie intelligent können Maschinen heute schon kommunizieren?

Es gibt bereits eine Vielzahl von Technologien, die Sprache erkennen und verarbeiten können, sei es schriftlich oder verbal. Diese Lösungen werden als Chatbots (auch Chatterbots genannt), Voicebots oder auch als (intelligente) Sprachassistenten bezeichnet. Bei diesen Systemen wird das gesprochene Wort des Nutzers regelbasiert analysiert, ausgewertet und auf Basis einer beschränkten Zahl definierter Prozesse werden Aktionen durchgeführt. Die Kommunikation mit einer Maschine ist bei diesen Sprachassistenten für den Benutzer jederzeit erkennbar. Anders soll es sich durch das Anlernen künstlicher Intelligenz in der Sprachverarbeitung verhalten, dem sogenannten Natural Language Processing (NLP). Mit NLP ist die Verarbeitung natürlicher Sprache gemeint, welche eine Disziplin im Bereich des maschinellen Lernens darstellt, die der Erforschung der Mensch-Maschine-Schnittstelle zugehörig ist [Pe16]. Da die KI ihren Wortschatz selbstständig erweitern kann, bedarf es keiner kompletten Definition jedes Wortes für eine natürliche Gesprächsführung, sondern lediglich ein Grundwortschatz und ein anschließendes Training, in dem die KI die Sprache selbst erlernt. Ziel ist es eine Maschine zu erschaffen, die sich intelligent verhält [Er16]. Um testen zu können, ob eine Maschine ein gleichwertiges Denkvermögen wie ein Mensch besitzt, formulierte Alan Turing im Jahr 1950 den Turing-Test. Bei diesem werden dem Tester zwei Partner gegenübergestellt, ein Programm und ein Mensch, mit denen er fünf Minuten via Text-Chat kommuniziert und beide ihn davon überzeugen sollen, dass sie menschlich sind. Wenn das Programm in 30 Prozent der Fälle als Mensch wahrgenommen wird, gilt der Test als bestanden. Es ist jedoch anzumerken, dass der Turing Test darauf abzielt menschliches Verhalten durch Maschinen nachzuahmen. Einen Gradmesser für die Lernfähigkeit von Programmen stellt dieser Test nicht dar. Erstmalig wurde dieser Test im Jahr 2014 durch den Chatbot „Eugen Goostmann“ bestanden, welcher sich als 13-jähriger ukrainischer Schüler ausgibt. [Ja14].

2.2 Chatbots

Chatbots, auch als Sprachassistenten, Voicebots oder digitale Assistenten bezeichnet, sind bereits seit Jahren auf dem Markt und ihre Möglichkeiten sind kontinuierlich gestiegen. Die Spracherkennung funktioniert mittlerweile sehr zuverlässig und die Kommunikation wirkt menschlich, da der Bot in einer menschenähnlichen Stimme antworten kann. Gute Beispiele und die bekanntesten Produkte im Bereich der Spracherkennung sind Apples Siri, Google Now oder auch Amazons Alexa. Alexa wird von Amazon als die dritte Generation in der Chatbot Technologie bezeichnet, welche zielorientiert arbeitet. So registriert der Chatbot, dass bspw. eine Flugbuchung gewünscht ist und führt daraufhin eine auf das Ziel ausgerichtete Konversation um alle Informationen zu bekommen, die für die Buchung notwendig sind [Az17]. Eine Branche, die die Entwicklungen im Bereich der digitalen Sprachassistenten besonders betrifft, ist die der Kommunikationsdienstleister. Speziell im Bereich der Callcenter existiert ein hohes Automatisierungspotenzial. Dies

betrifft zum einen die Backoffice-Prozesse, die durch Technologien wie Robotic Process Automation[2] automatisiert werden können als auch den 1st-Level Support, den vorrangig Callcenter-Agents übernehmen. Grundvoraussetzung für die Übernahme des 1st-Level Supports durch Sprachassistenten ist, dass diese in der Lage sind natürlich wirkende Gespräche zu führen (siehe nächsten Abschnitt in Bezug auf Google Duplex). Aufgrund der Weiterentwicklung der KI im Bereich der Sprachassistenten ist es nur eine Frage der Zeit bis dies der Fall ist. Für Betreiber von Callcentern hat dies überwiegend positive Effekte, da mit großen Einsparungen hinsichtlich der Personalkosten zu rechnen ist, als auch die aufwändige Ressourcenallokation wegfällt, da sich die digitalen Assistenten beliebig skalieren lassen und das Anfrageaufkommen somit flexibel bewältigen können. Aus diesem Grund wird auch eine höhere Servicequalität erwartet, aus der eine steigende Kundenzufriedenheit resultiert [KKK18].

Ein bekanntes Beispiel für einen intelligenten Chatbot aus dem Jahr 2016 ist der Versuch von Microsoft dem selbstlernenden Chatprogramm Tay die Kommunikation junger Menschen anzutrainieren, in dem er aus den Fragen und Antworten von Nutzern lernt. Der weibliche Avatar hatte Accounts auf Kik, GroupMe, Twitter, Facebook, Snapchat und Instagram um auf diesen Plattformen mit den Nutzern zu interagieren. Der Versuch ist jedoch gescheitert, da Tay bereits nach kurzer Zeit beleidigende und rassistische Tweets von sich gab und daraufhin von Microsoft abgeschaltet wurde. Ursache war der maschinelle Lernprozess, der auf dem basierte, was Nutzer geschrieben hatten [Be16]. Ein weiterer aktueller Anwendungsfall ist der KI Assistent Google Duplex, welcher auf der Google I/O am 08.05.2018 vorgestellt wurde und große Beachtung fand. Dieser kann im Gegensatz zu bisherigen digitalen Assistenten Telefonate für seinen Benutzer führen, bei denen, zumindest in den Demovideos von Google, nicht zwischen Mensch und Maschine unterschieden werden kann. So baut die KI kurze Denkpausen ein um natürlicher zu wirken und reagiert auch in komplexen Gesprächssituationen souverän [Kr18]. Im Kern von Duplex steckt ein rekurrentes neuronales Netzwerk (RNN), welches entwickelt wurde um den Anforderungen, die natürlich klingende Konversationen mit sich bringen, gerecht zu werden. Zur Umsetzung des RNN wurde die TensorFlow Extended (TFX) machine learning Plattform verwendet [LM18].

2.3 Textanalyse

Mithilfe von Text-Mining Verfahren, welche in der Lage sind Muster in Texten zu erkennen und diese zu klassifizieren, ist es unter anderem möglich die Stimmung in Texten zu analysieren. Als einer von mehreren Marktteilnehmern bietet Microsoft mit seinen Cognitive Services diverse Cloud-Funktionen an, die es bspw. ermöglichen einen Nutzer über negative oder positive Tweets zu informieren oder Texte auf die Stimmung hin zu untersuchen, so dass nach Analyse des betreffenden Textes ein Wert zwischen 0 und 100 zurückgegeben wird, wobei ein hoher Wert eine bessere Stimmung zum Ausdruck bringt

[2] Robotergesteuerte Prozessautomatisierung, die nur auf der Benutzeroberfläche arbeitet und keine technischen Schnittstellen benötigt. Vorrangig eingesetzt um einfache stark repetitive systemübergreifende Tätigkeiten zu automatisieren [KKK18].

[Mi17a]. Diese Funktionen können bspw. im Kundensupport eingesetzt werden um eine Vorauswahl bei der Bearbeitungsreihenfolge von eingehenden Mails zu treffen. Denkbar wäre hier, unzufriedene Kunden, deren Text einen niedrigeren Wert hat, bei der Bearbeitung höher zu priorisieren um schnell reagieren zu können und somit die Kundenbindung zu stärken. Ein weiterer Anwendungsfall wäre z. B. eine automatische Löschung negativer Kommentare unter Zuhilfenahme automatisierter Prozesse, die bei einer Unterschreitung eines bestimmten Wertes ausgeführt werden.

2.4 IBM Watson

Bei dem von IBM in Zusammenarbeit mit mehreren Universitäten entwickelten Software-Agenten „Watson“ handelt es sich um ein wissensbasiertes System, welches unter anderem eingesetzt wird um Fragen in natürlicher Sprache zu beantworten. Größere Bekanntheit erlangte Watson als es im Jahr 2011 in der US-Fernsehshow „Jeopardy“ gegen die zwei menschlichen Meister Brad Rutter und Kenn Jennings nach zwei Spielen gewann. Bedingt durch seine schnelle Reaktionszeit hatte Watson den Vorteil den Knopf für die Hupe schneller betätigen zu können und somit die erste Antwort auf die Fragen zu geben. Um diese Leistungsfähigkeit zu erreichen wurde Watson auf 90 IBM Power 750 Servern implementiert, von denen jeder mit 32 Prozessoren ausgestattet war [Er16]. Weitere Aufmerksamkeit bekam Watson, als die japanische Versicherungsfirma Fukoku Mutual Life Insurance 34 Angestellte durch die KI „IBM Watson Explorer“ ersetzt hat. Die KI soll Informationen aus Dokumenten heraussuchen und mit Unterstützung eines Algorithmus bspw. Vorschläge zu auszuzahlenden Beträgen machen. Die endgültige Entscheidung wird hier jedoch weiterhin ein Mensch treffen. Der japanische Versicherer verspricht sich durch die Einführung zum einen eine Reduzierung der Personalkosten um umgerechnet eine Mio. Euro sowie eine Produktivitätssteigerung von 30 Prozent [Pa17]. Mittlerweile gibt es jedoch auch negative Stimmen, so hat die Investmentbank Jefferies in einem Bericht für Investoren dargelegt, dass Watson für Aktionäre in absehbarer Zeit keine Vorteile bringen wird. Verwiesen wird unter anderem auf das Krebsforschungszentrum MD Anderson, welches 60 Mio. Dollar in die Technologie investiert hat ohne, dass sich die erhofften positiven Effekte eingestellt haben. Ein generelles Problem ist laut Jefferies, dass Werbeaussagen nicht eingehalten werden und die Deep Learning Funktionalitäten Defizite aufweisen [Ma17b].

2.5 Autonomes Fahren

Ein weiteres bedeutendes Anwendungsgebiet im Bereich der künstlichen Intelligenz ist das autonome Fahren. Die Entwicklung hin zum selbstfahrenden Fahrzeug geht stetig voran und aktuell gibt es bereits diverse Fahrassistenzsysteme, die Einparken können oder helfen die Spur zu halten. In Zukunft sollen selbstfahrende Autos, in der Literatur auch als „Roboterautos“ bezeichnet, Menschen autonom zum gewünschten Ziel bringen. Diese Autos wären auch in der Lage ohne Menschen zu fahren, wodurch vielfältige Einsatzszenarien möglich sind. Denkbar wäre bspw. das Absetzen des Fahrzeughalters am

Wunschort und die anschließende automatische Parkplatzsuche des Roboterautos. Auch in Bezug auf die Sicherheit ist mit einer deutlichen Verbesserung zu rechnen, so erwarten Experten, dass die Unfallzahlen um mehr als 90 Prozent zurückgehen werden [Er16]. Um den Stand der Technik einordnen zu können und einen Überblick über den Automatisierungsgrad zu bekommen, wurde in Europa und den USA eine Klassifizierung des autonomen Fahrens in sechs Stufen vorgenommen (Siehe Abb. 2-1). Beginnend bei 0, in der keine Automatisierung stattfindet geht die Skala bis zur Stufe 5, welche keinen menschlichen Fahrer benötigt.

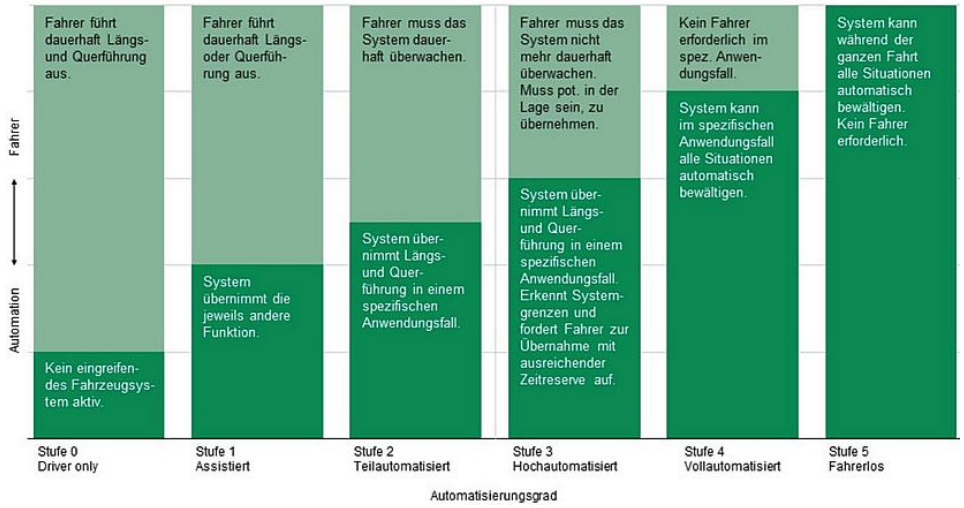


Abbildung 2-2: Stufen des automatisierten Fahrens [Sc17]

Ein aktueller Anwendungsfall für automatisiertes Fahren ist der Autopilot 2 von Tesla. Dieser verfügt über acht Kameras, eine rückwärtige sowie drei an der Frontscheibe und je zwei in B-Säule und Kotflügelblinker. Zusätzlich verfügt er über 12 Ultraschallsensoren für die Nahraumkontrolle und einen Radar [Sc17]. Anders als der Name vermuten lässt, handelt es sich hierbei nicht um Vollautomatisierung, sondern um eine Fahrassistenz, die sich zwischen Stufe 1 und 2 einordnen lässt. Genau diese Fehleinschätzung führte bereits zu mehreren teils folgenschweren Unfällen. So starb in Florida im Jahr 2016 ein Fahrer des Tesla Model S mit eingeschaltetem Autopilot bei einer Kollision mit einem ihm entgegenkommenden Truck, der links abbiegen wollte. In diesem Fall hätte eine Notbremsung oder ein Umlenkmanöver durch den Fahrassistenten eingeleitet werden müssen. Eine mögliche Ursache dafür ist laut Tesla, dass der Sattelzug weiß und der Himmel hell erleuchtet war, so dass das Bild der Kamera zu kontrastarm war. Des Weiteren könnte der Radar den LKW falsch klassifiziert und für eine Schilderbrücke gehalten haben. Nichtsdestotrotz ist der Fahrer bei solch einem teilautomatisierten System verpflichtet die Hände durchgehend am Lenkrad zu behalten um jederzeit eingreifen zu können, so dass dem Fahrer mindestens eine Mitschuld am Unfall zu geben ist [Ha16].

Eine weitere Hürde in Bezug auf das automatisierte Fahren stellen rechtliche Fragestellungen dar. Speziell in Bezug auf die Haftung bei der Unfallverursachung durch ein autonomes Fahrzeug kann die Ermittlung des Schuldigen schwierig werden. So müsste im Einzelfall ermittelt werden, ob der Halter, der Hersteller oder sogar der Software-Entwickler bzw. der für die Entwicklung beauftragte Dienstleister haftbar gemacht wird. Im Jahr 2017 hat die Bundesregierung diesbezüglich Regelungen für selbstfahrende Autos auf den Weg gebracht, deren Kern die Gleichstellung von Mensch und Computer ist. Hoch- oder Vollautomatisierte Systeme dürfen somit künftig das Fahren übernehmen. In Bezug auf die Haftung ist der Gesetzestext jedoch sehr unkonkret, da der „Fahrzeugführer (…) dazu verpflichtet (ist), die Fahrzeugsteuerung unverzüglich wieder zu übernehmen, wenn er erkennt oder aufgrund offensichtlicher Umstände erkennen muss, dass die Voraussetzungen für eine bestimmungsgemäße Verwendung der hoch- oder vollautomatisierten Fahrfunktionen nicht mehr vorliegen". Diese Formulierung lässt erkennen, dass die Haftung letztendlich beim Fahrzeughalter liegen dürfte [SP17].

2.6 AlphaGo und seine Nachfolger

Im Gegensatz zu Schachcomputern, die seit Jahren für Menschen praktisch unbezwingbar sind, gibt es noch viele weitere Herausforderungen für die KI. Ein Beispiel hierfür ist das japanische Spiel Go, welches auf einem quadratischen Brett mit 361 Feldern gespielt wird. Durch die hohe Komplexität (nach 8 Halbzügen ergeben sich etwa $1{,}5*10^{19}$ Stellungen) sind alle bekannten Spielbaum-Suchverfahren chancenlos gegen gute menschliche Spieler. Den ersten Sieg einer Maschine konnte im Januar 2016 das Programm AlphaGo von Google DeepMind für sich verbuchen, als es den Europameister Fan Hui mit 5:0 schlug [Er16]. Eine Weiterentwicklung von AlphaGo stellt AlphaGo Zero dar, welches im Gegensatz zu AlphaGo mit keinerlei Vorwissen über das Spiel, sondern ausschließlich mit den Spielregeln ausgestattet gegen sich selbst trainiert. Diese Version war bereits nach 40 Tagen Training stärker als die vormals stärkste AlphaGo Version mit dem Namen Master. Der gesamte Prozess von AlphaGo Zero basiert auf einem Lernalgorithmus, der mit einem Suchsystem kombiniert wird. Dies bedeutet, dass AlphaGo Zero aus Versuchen und Fehlern lernt und mit diesem Suchsystem jede ihm mögliche Bewegung selbst herausfindet. Nach jedem Spiel erhielt das System Feedback, +1 für Sieg und -1 für eine Niederlage. So konnte das neuronale Netzwerk hinter AlphaGo Zero sich nach jedem Spiel automatisch in eine (theoretisch) bessere Version konfigurieren. In Bezug auf die Hardware kam AlphaGo Zero auch mit weniger Ressourcen als seine Vorgänger aus. Kamen in den vorherigen Versionen noch 48 Tensor Processing Units zum Einsatz waren es jetzt nur noch vier [Bu17]. Ende 2017 konnte Google DeepMind erneut für Aufsehen sorgen, nachdem die leistungsfähigere Software AlphaZero die führenden Programme beim Schach, Shogi und Go besiegen konnte. Die Besonderheit von AlphaZero liegt darin, das es nur die Grundregeln der Spiele gezeigt bekommt und dann mithilfe von selbstlernenden Algorithmen und großer Rechenkraft gegen sich selbst spielt und aus den Fehlern lernt. AlphaZero hatte bereits nach 4 Stunden das Schachspiel so gut gelernt, dass es die führende Software Stockfish bezwingen konnte. Auch beim Go konnte bereits nach 8 Stunden Training die aktuellste Version von AlphaGo bezwungen werden [Br17].

2.7 Bilderkennung

Die Bilderkennung stellt ein bedeutendes Forschungsfeld im Bereich der künstlichen Intelligenz dar und einige Funktionen haben bereits ihren Weg in den Alltag gefunden. So können Gesichter seit einigen Jahren auf Bildern erkannt und Personen identifiziert werden. Auch Smartphones lassen sich mittlerweile per Gesichtserkennung entsperren. Im Rahmen der intelligenten Gesichtserkennung besteht eine große Herausforderung darin einem System beizubringen Bildelemente richtig zu erkennen. Hierfür ist die Analyse von Millionen von Bildern als Trainingsdaten nötig, auf denen das zu erkennende Objekt oder Lebewesen zu sehen ist [Ma17a]. Microsoft bietet im Rahmen seiner Cognitive Services Schnittstellen zum maschinellen Sehen an, die bspw. eine automatische Beschriftung für das jeweilige Bild generieren, Gegenstände erkennen oder auch Videos automatisch untertiteln, in dem das auf dem Video zu sehende beschrieben wird [Mi17b]. Auch Facebook bietet ähnliche Funktionen zur Beschreibung von Bildern und zur Gesichtserkennung an [Ho16].

2.8 Robotik

In Bezug auf physische Roboter im Alltag haben bereits die ersten Geräte Einzug in unsere Haushalte gehalten. So sind Mäh- oder Staubsauerroboter auf dem Markt, die nach Einweisung (abstecken der Rasenfläche etc.) ihre Arbeit selbstständig verrichten und für den Nutzer eine Hilfe im Haushalt sind. In Zukunft sollen intelligente Serviceroboter diverse weitere Tätigkeiten übernehmen können. Neben dem Einsatz im Haushalt sind zudem viele weitere Einsatzgebiete denkbar, wobei der Einsatz im Gesundheitswesen als auch der militärische Einsatz von Kampfrobotern in Krisengebieten von besonderem Interesse ist [Er16]. In Japan und Südkorea sind bereits vermehrt emotionale Roboter im Einsatz, die in direkten Kontakt mit Patienten und Pflegebedürftigen kommen. Ein Beispiel hierfür ist der humanoide 1,20m große Roboter „Pepper“, der von der französischen Firma Aldebaran und dem japanischen Softbank-Konzern entwickelt wurde [Ba17]. Dieser kann unter anderem Gesichter erkennen, Sprache verstehen und Befehle entgegennehmen. Eine intelligente Sprachverarbeitung ist jedoch noch nicht möglich, so dass er noch keine menschenähnlichen Dialoge führen kann [Pr17a].

2.9 Militär

Wie bereits im vorigen Abschnitt erwähnt ist der militärische Einsatz von Robotern von besonderem Interesse. Der Einsatz von KI beim Militär ist jedoch nicht ausschließlich im Bereich der Roboter interessant, sondern hat bereits Einzug in diverse Waffensysteme gehalten. Viel diskutiert werden autonome Waffensysteme, die eigenständig und ohne menschliche Unterstützung operieren. Das amerikanische Militär hat eine nur wenige Zentimeter große Drohne Namens „Perdix“ entwickelt, die in der Lage ist im Schwarm anzugreifen [Fr17]. Bereits im Einsatz sind patrouillierende halbautonome Roboter in der demilitarisierten Zone zwischen Nord- und Südkorea. Der von der Samsung-Tochter

Techwin entwickelte Roboter vom Typ SGR-A1 ist mit einem Maschinengewehr ausgerüstet und soll mittels Bewegungssensoren und Wärmebildkamera in der Lage sein Feinde über 4km zu entdecken und automatisch zu feuern [Lo17]. Diese Art der Kriegsführung ist stark umstritten. So haben sich führende KI Wissenschaftler zusammengetan und ein Verbot autonomer Waffensysteme gefordert [Fr17].

3 Gesellschaftliche Auswirkungen

Es ist zu erwarten, dass der Trend hin zur Automatisierung von menschlicher Arbeit weiter anhält und einen großen Einfluss auf den Arbeitsmarkt haben wird. Betroffen sind bspw. in Bezug auf das autonome Fahren Taxi- und LKW-Fahrer oder auch Call-Center Agents hinsichtlich der Weiterentwicklung von Chatbots. So wird in einer Studie von PricewaterhouseCoopers LLP prognostiziert, dass in Deutschland bis zum Jahr 2030 35 Prozent der Arbeitsplätze einem potenziell hohen Risiko unterliegen, automatisiert zu werden. Der größte Arbeitsplatzrückgang wird in den Branchen Transport, Groß- und Einzelhandel sowie produzierendes Gewerbe erwartet [Pr17b]. Es entstehen zwar neue Arbeitsplätze, beispielsweise in der Überwachung und Administration der eingesetzten Techniken, jedoch erfordern diese Tätigkeiten andere Qualifikationen als die zuvor weggefallenen Tätigkeiten, so dass Menschen ohne spezielle Qualifikation nur schwer in den Arbeitsmarkt reintegriert werden können [KKK18]. Neben den Auswirkungen auf den Arbeitsmarkt existieren auch zahlreiche ethische Bedenken in Bezug auf den Einsatz von KI. Die Gesichtserkennung ist heute schon sehr zuverlässig und unter Zuhilfenahme von KI bei der Analyse von Überwachungsvideos ist es möglich Personen zu identifizieren und Bewegungsprofile zu erstellen. Der militärische Einsatz von KI wird ebenfalls stark kritisiert, da hier Maschinen über Leben und Tod entscheiden. Dieser Sachverhalt ist auch hinsichtlich des autonomen Fahrens von Bedeutung. So stellt sich die Frage wie ein autonomes Fahrzeug reagiert, wenn bspw. ein Unfall unausweichlich ist und entschieden werden muss welchen Verkehrsteilnehmer ausgewichen wird. Durch den Einsatz von KI ist eine deutliche Produktivitätssteigerung zu erwarten, die, wenn die Produktivitätsgewinne gerecht verteilt werden, wachsenden Wohlstand für jeden bedeuten könnte. Es ist jedoch wahrscheinlicher, dass von diesen Gewinnen nur wenige profitieren werden, nämlich die Kapitaleigner, vorwiegend wenige Reiche und Banken [Er16].

Literaturverzeichnis

[Az17] Azoff, M.: 2017 Trends to Watch: Artificial Intelligence. The impact of deep learning in verticals and Internet of Things, 2017.

[Ba17] Balzter, S.: Künstliche Intelligenz: Wie Roboter im Gesundheitswesen eingesetzt werden können. http://www.faz.net/aktuell/wirtschaft/diginomics/wie-roboter-im-gesundheitswesen-eingesetzt-werden-koennen-15336071.html,

10.05.2018.

[Be16] Beuth, P.: Microsoft: Twitter-Nutzer machen Chatbot zur Rassistin. https://www.zeit.de/digital/internet/2016-03/microsoft-tay-chatbot-twitter-rassistisch, 09.05.2018.

[Br17] Breithut, J.: Künstliche Intelligenz AlphaZero: In vier Stunden zum Schachweltmeister. http://www.spiegel.de/netzwelt/web/google-ki-alphazero-meistert-schach-und-go-a-1182395.html, 07.05.2018.

[Bu17] Burgess, M.: Es gibt nur noch einen Gegner für Googles KI AlphaGo. https://www.wired.de/collection/science/googles-ki-alphago-lernt-von-alphago, 07.05.2018.

[Er16] Ertel, W.: Grundkurs Künstliche Intelligenz. Eine praxisorientierte Einführung. Springer Fachmedien Wiesbaden; Imprint: Springer Vieweg, Wiesbaden, 2016.

[Fr17] Freidel, M.: Autonome Waffen: Wenn Maschinen Krieg führen. http://www.faz.net/aktuell/politik/inland/autonome-waffen-wenn-maschinen-krieg-fuehren-15202125.html, 10.05.2018.

[Ha16] Harloff, T.: Warum Teslas Autopilot beim tödlichen Unfall irrte. http://www.sueddeutsche.de/auto/selbstfahrendes-auto-warum-teslas-autopilot-beim-toedlichen-unfall-irrte-1.3059246, 09.05.2018.

[Ho16] Holland, M.: Barrierefreiheit: Facebook lässt KI Bilder beschreiben. https://www.heise.de/newsticker/meldung/Barrierefreiheit-Facebook-laesst-KI-Bilder-beschreiben-3162331.html, 10.05.2018.

[Ja14] Janschitz, M.: KI besteht Turing-Test: Darum ist der Hype unbegründet [Kommentar]. https://t3n.de/news/turing-test-ki-eugene-goostman-549886/, 07.05.2018.

[KKK18] Kharchenko, A.; Kleinschmidt, T.; Karla, J.: Callcenter 4.0 – Wie verändern Spracherkennung, Künstliche Intelligenz und Robotic Process Automation die bisherigen Geschäftsmodelle von Callcentern. In HMD Praxis der Wirtschaftsinformatik, 2018.

[Kr18] Kremp, M.: Google Duplex auf I/O: Gruselig gute künstliche Intelligenz. http://www.spiegel.de/netzwelt/web/google-duplex-auf-der-i-o-gruselig-gute-kuenstliche-intelligenz-a-1206938.html, 10.05.2018.

[LM18] Leviathan, Y.; Matias, Y.: Google Duplex: An AI System for Accomplishing Real-World Tasks Over the Phone. https://ai.googleblog.com/2018/05/duplex-ai-system-for-natural-conversation.html, 10.05.2018.

[Lo17] Lobe, A.: Drohnen als Waffensysteme: Tod durch Roboter. http://www.faz.net/aktuell/feuilleton/debatten/drohnen-als-waffensysteme-

tod-durch-roboter-15327836.html, 10.05.2018.

[Ma17a] Martin-Jung, H.: So trainiert Google künstliche Intelligenz. http://www.sueddeutsche.de/digital/algorithmen-so-trainiert-google-kuenstliche-intelligenz-1.3759233, 10.05.2018.

[Ma17b] Marwan, P.: IBM Watson: Vorzeigetechnolgie mit ersten Macken. https://www.silicon.de/41653615/ibm-watson-vorzeigetechnolgie-mit-ersten-macken/, 09.05.2018.

[Mi17a] Microsoft Corporation: Textanalyse | Microsoft Flow. https://flow.microsoft.com/de-de/connectors/shared_cognitiveservicestextanalytics/textanalyse/, 08.05.2018.

[Mi17b] Microsoft Corporation: Maschinelles Sehen. https://azure.microsoft.com/de-de/services/cognitive-services/computer-vision/#analyze, 10.05.2018.

[Pa17] Patrick Welter: Japan: Versicherer ersetzt zahlreiche Mitarbeiter durch künstliche Intelligenz. http://www.faz.net/aktuell/wirtschaft/japan-versicherer-ersetzt-mitarbeiter-durch-ki-ibm-watson-14605854.html, 25.04.2017.

[Pe16] Petereit, D.: Was ist eigentlich der Unterschied zwischen AI, Machine Learning, Deep Learning und Natural Language Processing? https://t3n.de/news/ai-machine-learning-nlp-deep-learning-776907/, 09.05.2018.

[Pr17a] Propson-Hauck, M.: Humanoider Roboter Pepper: „Du Mistkerl, warum machst Du das nicht?“. http://www.faz.net/aktuell/rhein-main/wirtschaft/ics-entwickler-statten-roboter-pepper-mit-digitalem-leben-aus-14887829.html, 10.05.2018.

[Pr17b] PricewaterhouseCoopers LLP (PwC): UK Economic Outlook, o. O., 2017.

[Ri88] Rich, E.: Artificial intelligence. McGraw-Hill, Auckland [u.a.], 1988.

[Sc17] Schwarzer, C. M.: Hochautomatisiertes Fahren: Ein Zwischenstandsbericht. https://www.heise.de/autos/artikel/Hochautomatisiertes-Fahren-Ein-Zwischenstandsbericht-3607672.html, 09.05.2018.

[SP17] SPIEGEL ONLINE GmbH: Autonomes Fahren: Regierung beschließt Autopilot-Gesetz. http://www.spiegel.de/auto/aktuell/autonomes-fahren-regierung-beschliesst-autopilot-gesetz-a-1131675.html, 10.05.2018.

Empfehlungssysteme mit Bezug auf den Praxispartner Brille24

Jonas Tiemerding[1]

Abstract: Recommender Systems are techniques to suggest items which may be interesting for an individual user. This paper introduces different filters of Recommender Systems. The differences between collaborative, content-based, demographic, utility-based, knowledge-based, community-based and hybrid filters are explained. In chapter 3 recommender engines are expanded by context. When you use context-aware recommender systems the two-dimensional model of recommender system turns into a multi-dimensional model. Every aspect of content creates another dimension. At the end of the chapter it is explained what means context for Brille24. Chapter 4 introduces the recommender engine of Brille24. It is explained which recommender engine they use, how they can influence the results and how they benefit from the use of their recommender engine. At the end problems and limits of recommender engines are listet.

Keywords: Recommender Systeme; Empfehlungssysteme; kontextsensitiv; Brille24

1 Einführung

Da die Fülle an Informationen für einen Benutzer schnell unübersichtlich werden kann, wird durch Recommender Systeme versucht, dem Nutzer Empfehlungen auf Grundlage seiner Interessen zu bieten. Diese berechnen sich beispielsweise aus Produktbewertungen, die er zuvor abgegeben hat, oder aus Dingen, die er gekauft hat. Jedoch berücksichtigen diese oft nur eine allgemeingültige Situation und beziehen sich nicht auf spezifische Begebenheiten. Daher wird auch auf den Kontext des Nutzers eingegangen. Durch eine kontextsensitive Filterung der Empfehlungen wird versucht, dem Nutzer noch spezifischere Empfehlungen auszusprechen. Dies ist insofern sinnvoll, als dass sich die Ergebnisse stark von denen einer normalen Filterung unterscheiden. Besonders bei einem breiten Informationsspektrum führt dies zu Ergebnissen mit einer höheren Relevanz. Recommender Systeme spielen heute schon eine wichtige Rolle und lassen sich auf vielen Plattformen in unterschiedlichen Ausführungen beobachten. Mögliche Einsatzgebiete sind beispielsweise im E-Commerce oder auf Streaming Plattformen. Besonders auf großen und populären Internetseiten sowie Streaming-Diensten lassen sich Recommender Systeme finden, was die Bedeutung dieser Technik unterstreicht. Beispiele hierfür sind Amazon, Netflix, YouTube, IMdB, Spotify oder aber auch beim Praxispartner Brille24. Da die Fülle an Informationen in Zukunft

[1] CvO Universität Oldenburg, VLBA, Hundsmühler Straße 74, 26131 Oldenburg, Deutschland, jonas.tiemerding@uni-oldenburg.de

noch weiter ansteigen wird, spielen Empfehlungssysteme eine immer bedeutendere Rolle zur Filterung relevanter Informationen. Recommender Systeme stellen bei neu angelegten Profilen insofern ein Problem dar, als dass sie nur schlecht ohne zur Verfügung stehende Daten agieren können. Deswegen lassen sich zu Beginn nur schwierig personalisierte Empfehlungen aussprechen. In dieser Arbeit werden auch Modelle vorgestellt die versuchen dieses Problem zu umgehen.

1.1 Was sind Recommender Systeme?

Recommender Systeme sind Empfehlungssysteme, die einem Benutzer Vorschläge für eine bestimmte Tätigkeit liefern. Beispielsweise schlagen sie dem Nutzer vor, welchen Film er sich anschauen sollte, oder welches Buch er lesen sollte. Sie sollen dem Benutzer bei der Entscheidungsfindung helfen und seine Suchzeiten verkürzen. Der Objekttyp, der dem Benutzer empfohlen werden soll, wird hierbei als Item deklariert. Um möglichst relevante Empfehlungen zu liefern, operieren Recommender Systeme meistens für jeden Nutzer individuell. Nicht personalisierte Empfehlungen sind zum Beispiel Bestsellerlisten oder Musik-Charts, wie sie oft in Zeitschriften oder Tageszeitungen abgedruckt werden. Auch diese können ihren Zweck erfüllen und hilfreich sein, sind aber in der Regel nicht das Ergebnis aus Recommender Systemen. In ihrer einfachsten Form sind personalisierte Recommender Systeme als sortierte Liste aufgebaut. Dabei erfolgt die Sortierung nach größtmöglicher Übereinstimmung mit den Interessen des Nutzerprofils. Zur Berechnung von Vorschlägen werden in der Regel Nutzerprofile oder Items miteinander verglichen. Wurden ausreichend viele Ähnlichkeiten zwischen diesen Objekten erkannt, kann auf dieser Grundlage eine Empfehlung abgegeben werden. Nutzerpräferenzen können dabei entweder direkt erfasst werden, zum Beispiel durch Bewertungen oder indirekt, indem sie aus dem Verhalten des Nutzers gefolgert werden. Zum Beispiel kann der Besuch einer Produktseite als Zeichen des Interesses interpretiert werden [RRS11].

2 Filterarten

Im Folgenden werden zunächst die kollaborativen und die inhaltsbasierten Filter ausführlich erläutert und miteinander verglichen. Anschließend werden weitere Filtermethoden vorgestellt.

2.1 Kollaborative Filter

Kollaboratives Filtern (collaborative filtering): Beim Kollaborativen Filtern werden Empfehlungen gegeben, indem ähnliche Nutzer miteinander verglichen werden. Anhand der Bewertungen von Nutzern mit ähnlichen Interessen werden Empfehlungen für den Benutzer

ausgesprochen. Hierfür müssen zunächst die Ähnlichkeiten von Nutzerprofilen erfasst werden. Aus diesen Daten können nun Rückschlüsse auf möglicherweise interessante Objekte für den User geschlossen werden. Dieses Modell wird am häufigsten verwendet. [Bu02]

Zur Umsetzung von kollaborativen Filtern gibt es verschiedene Ansätze: Memory-based kollaborative Filter, Model-based kollaborative Filter und hybride Empfehlungen. Hybride Filter werden im Unterkapitel 2.4 erläutert.
Bei Memory-based kollaborativen Filtern, also speicherbasierten Filtern, wird die ganze oder Teile der Benutzerdatenbank genutzt. Dabei werden Nutzer in Gruppen mit dem gleichen Interesse unterteilt. Zu jedem Benutzer werden sogenannte Nachbarn erstellt, die ähnliche Interessen haben und auf deren Grundlage Empfehlungen ausgesprochen werden können. Um Ähnlichkeiten zwischen zwei Items feststellen zu können, werden in der Regel Daten von Nutzern erhoben, die schon viele Items bewertet haben. Danach wird versucht diese verschiedenen Items in Verbindung zu setzten: Benutzern denen Item X gefällt, gefällt auch häufig Item Y oder Benutzer die Item X positiv bewertet haben, haben Item Z negativ bewertet.
Die Vorteile dieser Methode sind sowohl die leichte Implementation aber auch die leichte Erweiterbarkeit, wenn neue Items hinzukommen. Außerdem muss der genaue Inhalt der Items nicht betrachtet werden. Probleme treten auf, wenn es nicht ausreichend Daten gibt. Stehen nur wenige Daten zur Verfügung wird es für den Algorithmus schwieriger passende Items zu empfehlen. Ein weiteres Problem besteht darin, dass der Algorithmus von menschlichen Bewertungen abhängig ist. Bei neu angelegten Benutzerkonten können außerdem nur schwierig treffende Empfehlungen ausgesprochen werden, da diese wenig Daten bereitstellen.

Bei model-based kollaborativen Filtern wird versucht durch Machine-Learning und Data-Mining Algorithmen komplexe Muster zu ermitteln. Dabei werden die Algorithmen erst durch Testdaten trainiert um dann anschließend Empfehlungen mittels der Realdaten auszusprechen. Diese verschiedenen Modelle wurden entwickelt, um die Nachteile des memory-based Filtering zu umgehen. Sie liefern bessere Ergebnisse, wenn nur wenige Daten zur Verfügung stehen. Der Nachteil hierbei ist die sehr kostenintensive Modellerstellung.[SK09]

2.2 Inhaltsbasierte Filter

Inhaltsbasierte Filter (content-based filtering): Das Inhaltsbasierte Filtern vergleicht im Gegensatz zum kollaborativen Filtern nicht User miteinander, sondern Objekte. Die Bewertungen der Objekte eines einzelnen Users führen dazu, dass für diesen User ähnliche Objekte gesucht und vorgeschlagen werden. Die Profile und Bewertungen anderer Nutzer haben also keinen Einfluss auf die Empfehlungen. [Bu02]

Zur Umsetzung dieses Ansatzes werden den Objekten verschiedene Eigenschaften

entzogen. Ein Film kann beispielsweise nach den Eigenschaften Regisseur, Schauspieler, Erscheinungsjahr, Genre und Thema aufgeteilt werden. Nach der Einteilung der Eigenschaften von verschiedenen Filmen, wird geschaut welche Eigenschaften Filme erfüllen, die besonders häufig gute Bewertungen vom Nutzer erhalten haben. Als Empfehlung werden weitere Filme mit einer hohen Übereinstimmung der als positiv empfundenen Eigenschaften abgegeben.
Damit dieser Ansatz funktionieren kann, muss der Benutzer allerdings erst einige Filme bewertet haben, da ansonsten keine Datengrundlage vorhanden ist. [AT05]

2.3 Vergleich zwischen inhaltsbasierten und kollaborativen Filtern

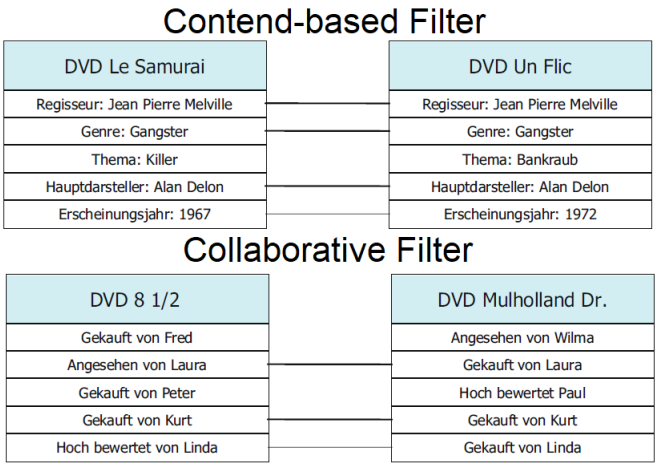


Abb. 1: Vergleich zwischen conetent-based und collaborative Filtering [MS12]

Abbildung 1 veranschaulicht die beiden unterschiedlichen Konzepte. Oben ist das Konzept des conenent-based Filtering dargestellt. Die beiden Filme „Le Samurai“ und „Un Flic“weisen viele gleiche Eigenschaften auf. Daher ist es wahrscheinlich, dass wenn man sich für einen dieser Filme interessiert oder diesem eine gute Bewertung gegeben hat, der andere als Empfehlung ausgesporchen wird. Hierbei sind die Eigenschaften verschieden zu gewichten. Der Regisseur lässt sich vielleicht nicht gut mit anderen Regisseuren vergleichen. Daher macht es Sinn, wenn es sich um denselben handelt. Beim Erscheinungsjahr ist es hingegen vielleicht nicht relevant ob der andere Film aus dem selben Jahr stammt. Hierbei wäre eine Unterteilung in Zeiträume aber eventuell sinnvoll.
Unten ist das Konzept des collaborative Filtering dargestellt. Die Filme „8 1/2“ und „Mullholland Dr.“ wurden beide von Kurt gekauft, von Linda hoch bewertet oder gekauft und von Laura angesehen oder gekauft. Hat der User nun Interessen wie Kurt, Laura oder Linda und kann somit mit diesen Nutzern verglichen werden, können ihm auf dieser Grundlage die Filme empfohlen werden. [MS12]

2.4 Weitere Filtermethoden

Robin Burke stellt in einem Artikel verschiedene Arten von Empfehlungssystemen vor [Bu02]. Neben dem collaborative und content-based filtering erläutert er demographische, nutzenbasierte und wissensbasierte Filter. Ricci u. a. [RRS11] stellen zudem noch zwei weitere Techniken vor: Gruppenorientierte und Hybride Recommender Systeme.

Demographisches Filtern (demographic filtering): Bei Demographischen Empfehlungssystemen werden Benutzer mit vergleichbaren Eigenschaften in Gruppen eingeteilt. Die Einteilung erfolgt beispielsweise nach Alter, Berufsstand oder Wohnort.

Nutzenbasiertes Filtern (utility-based filtering): Beim Nutzenbasierten Filtern werden Nutzer nach ihren geäußerten Bedürfnissen in Gruppen unterteilt. Die Berechnung erfolgt also nach dem Nutzen des Objekts für den Benutzer.

Wissensbasiertes Filtern (knowledge-based filtering): Wissensbasierte Filter erstellen Bedarfsprofile zu den Nutzern und leiten somit aus vorangegangenen Bedürfnissen den Nutzen von Objekten für den Benutzer ab.

Gruppenorientiertes Filtern (community-based filtering): Die Empfehlungen werden hier aus den Präferenzen der Freunde des Users generiert. Bei diesem Ansatz wird davon ausgegangen, dass man den Empfehlungen von Freunden eher Folge leisten würde, als denen von Fremden, aber vergleichbaren Nutzern. Dieser Ansatz lässt sich vor allem aus dem enormen Erfolg von sozialen Netzwerken begründen.

Hybride Recommender Systeme (hybrid recommender Systems): Bei Hybriden Recommender Systemen handelt es sich um eine Kombination der vorgestellten Filterungsansätze. Durch die Kombination von verschiedenen Techniken wird versucht, die Vorteile der verschiedenen Methoden so zu nutzen, dass sie ihre Schwächen gegenseitig beheben. [HK07]

3 Kontextsensitive Recommender Systeme

Kontextsensitive Recommender Systeme beziehen sich auf einen individuellen Kontext, indem sich der Benutzer derzeit befindet. Ist der Kontext bekannt, können dem Nutzer bessere Empfehlungen ausgesprochen werden. In diesem Kapitel wird zunächst einmal erläutert, was Kontext ist und was ein kontextsensitives Recommender System ausmacht. Anschließend wird das dadurch entstehende multidimensionale Modell kurz erläutert. Am Ende wird der Kontext mit Bezug auf Brille24 beschrieben und Möglichkeiten vorgestellt, wonach gefiltert werden kann.

3.1 Was ist Kontext

Die englischsprachige Standard-Definition von Kontext lautet: „conditions or circumstances which affect some thing“ [AT11]. Kontext ist also eine Bedingung oder ein Umstand, der eine beliebige Sache beeinflusst. In dieser Arbeit wird der Begriff Kontext im Zusammenhang mit Empfehlungssystemen gesehen. Eine Filterung nach dem Kontext macht insofern Sinn, als das die Empfehlungen noch präziser auf die Situation in der sich der Nutzer befindet ausgesprochen werden können. Ein Empfehlungssystem für Filme kann beispielsweise am Wochenende andere Filme empfehlen als an einem Arbeitstag oder bei Regen und Sonnenschein verschiedene Empfehlungen aussprechen. Voraussetzung hierfür ist oftmals, dass die Informationen über den Kontext dem Recommender System überhaupt zur Verfügung stehen.

3.2 Vom zweidimensionalen zum multidimensionalen Modell

In Recommender Systemen wird ein zweidimensionales Modell verwendet, aus dem die Empfehlungen errechnet werden. Die Bewertung (R: Rating) besteht aus den zwei Dimensionen - Benutzer (U: User) und Objekt (I: Item). Dadurch entsteht die Funktion:

$$R : U \times I \rightarrow R \tag{1}$$

Wird dieses Modell nun durch den Kontext erweitert, so entsteht aus dem zweidimensionalen Modell ein multidimensionales. Der Kontext kann dabei beliebig viele Ebene annehmen. Beispielsweise würde ein Empfehlungssystem für Spielfilme die Zeit (Wochentag oder Wochenende), den Ort (zu Hause oder im Kino) und die Gesellschaft (alleine, mit FreundIn oder mit Familie) als zusätzliche Dimensionen mit einbeziehen. Die entsprechende durch den Kontext (C: Context) erweiterte Funktion hierfür sieht wie folgt aus:

$$R : U \times I \times C \rightarrow R \tag{2}$$

In Abbildung 2 wird das multidimensionale Modell unter dem Kontext Zeit veranschaulicht. Die Dimension User besteht aus UserID, UserName und UserAlter. Die Dimension Item besteht aus ItemID, ItemName und ItemKosten und die Dimension Zeit besteht aus TimeID und TimeName. [AT11]

3.3 Kontext in Bezug auf Brille24

Auch bei der Brille24 könnten durch den Kontext der Nutzer bessere Empfehlungen ausgestellt werden. Hierbei sind verschiedene Szenarien denkbar. Die Zeit kann in verschiedenen

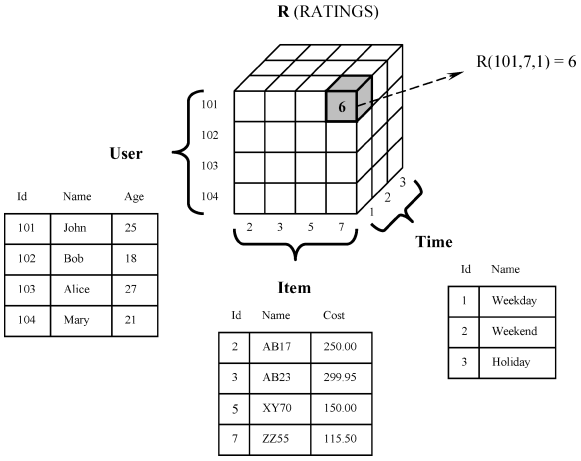


Abb. 2: Multidimensionales Modell [AT11]

Dimensionen Auswirkungen auf den Kauf haben. So kann beispielsweise nach der Jahreszeit unterschieden werden. Hierbei ist anzunehmen, dass die Nachfrage nach Sonnenbrillen im Frühjahr höher ist als im Herbst. Aber auch der Wochentag oder sogar die Tageszeit kann den Kauf beeinflussen. Weiter wichtige Merkmale sind demographische Daten, wie zum Beispiel Geschlecht und Alter des Kunden. Auch Geo-Daten können relevant sein. Besonders wenn sich eine Aussage über die Einkommensstruktur in bestimmten Gebieten oder Stadtteilen vornehmen lässt. So können in Einkommensstarken Gegenden auch kostenintensivere Brillen angeboten werden. Des Weiteren können auch Informationen erlangt werden, indem die Kampagnen, über die die Kunden auf das Produkt aufmerksam geworden sind, betrachtet werden.

4 Recommender Systeme bei Brille24

Im Folgenden wird beschrieben, welche Recommender Engine bei der Brille24 derzeit eingesetzt wird. Dazu wurde ein Interview mit Herrn Gebken von der Brille24 geführt. Das Gespräch wird nur inhaltlich, jedoch nicht in Interviewform dargestellt.

Die Brille24 verwendet für ihren Webshop eine Recommender Engine des externen Anbieters Prudsys. Abbildung 3 zeigt das Empfehlungssystem im Web-Shop der Brille24 (www.Brille24.de). Die Empfehlungen erscheinen sobald auf der Hauptseite eine Brille ausgewählt wurde. Sie sind unter dem ausgewählten Produkt am Ende der Seite aufgeführt. Die Software stellt eine reine Recommender Engine dar, die auf java basiert. Da hierbei hohe Geschwindigkeiten gefragt sind, ist die Engine Datenbank-los. Die genaue Funktionalität des Algorithmus ist für die Brille24 nicht ersichtlich, da sich die Recommenadation Engine durch den externen Anbieter in einer Blackbox befindet. Das Empfehlungssystem filtert

inhaltsbasiert. Das heißt, in dem Moment, wo der Nutzer die Seite aufruft, kann noch keine individuelle Empfehlung ausgesprochen werden. Erst das Klicken auf verschiedene Brillenmodelle liefert dem Empfehlungssystem Informationen. Umso mehr Brillen angeklickt wurden, desto besser kann der Algorithmus eine Empfehlung aussprechen. Bei Kunden mit bereits bestehendem Benutzerkonto können Empfehlungen aufgrund ihrer Sales History gemacht werden. Hierbei wird eine Brille die gekauft wurde deutlich mehr gewichtet, als eine Brille die nur angeschaut wurde. Umso tiefer der Kunde im Bestellvorgang vorangeschritten ist, desto mehr Auskunft gibt dies über seine Interessen. Auch dieser Aspekt wird von der Recommeder Engine einbezogen.
Die Brille24 hat die Möglichkeit Einfluss auf die Ergebnisse zu nehmen. Sie kann beispielsweise beliebige Parameter verändern und somit bestimmte Produkte in den Vordergrund rücken. Es können verschiedene Items, Listen oder auch Kategorien konfiguriert werden. Durch die manuelle Einflussnahme können die Ergebnisse der Engine aber derart verändert werden, dass nicht mehr die gewünschten Objekte empfohlen werden, da der Algorithmus für die Brille24 nicht einsehbar ist und somit seine Funktionsweise nicht genau nachvollzogen werden kann.

Abb. 3: Empfehlungen von Brille24

Unter den Top-Empfehlungen landen sehr oft die Brillen die auch insgesamt am häufigsten verkauft werden. Da bei ihnen ein Kauf am wahrscheinlichsten scheint. Die angewendete Recommender Engine kontrolliert sich dabei selbst, indem in regelmäßigen Abständen eine Brille, die kein Verkaufsschlager ist unter den Top-Empfehlungen ausgesprochen wird. Dadurch soll überprüft werden, ob die Top-Empfehlungen nur auf den ersten Plätzen stehen, da die Recommender Engine diese vorschlägt. Wird nun ein unbeliebtes Produkt deutlich mehr verkauft, weil es als Empfehlung ausgesprochen wurde, ist dies ein Zeichen dafür, dass es nicht relevant ist was als Empfehlung ausgesprochen wird, solange überhaupt Empfehlungen ausgesprochen werden.

Rund 11% der Käufer auf der Seite haben den Kauf unmittelbar durch die Empfehlung getätigt. Dies ist als sehr guter Wert zu verstehen und rechtfertigt den die Anwendung der Recommender Engine. Jedoch geht die Brille24 davon aus, dass tatsächlich nur 3-4% den Kauf aufgrund des Empfehlungssystem getätigt haben, die anderen 7-8% hätten auch ohne Empfehlung eine Brille gekauft. Im Allgemeinen ist die Brille24 derzeit sehr zufrieden mit den Ergebnissen der Recommender Engine. Jedoch könnten sich die Ergebnisse durch die Betrachtung des Kontext stark verbessern. Dafür müsste die Brille24 von der bisherigen

Recommender Engine abrücken.

In Newslettern wird aktiv mit den individuellen Top-Empfehlungen geworben. Laut Einschätzung von Herrn Gebken sind die Kunden deutlich offener für Empfehlungssysteme geworden. Vor allem in Deutschland standen solchen Systemen oft Datenschutzbedenken der Kunden im Wege. Der technologische Fortschritt und die Notwendigkeit Empfehlungen auszusprechen, da sich die Kunden bei einem Überangebot sonst nicht mehr zurecht finden, stellt die Datenschutzbedenken immer mehr in den Hintergrund.

5 Probleme und Grenzen

Recommender Systeme haben allgemein das Problem, dass besonders bei neu angelegten Benutzerprofilen oft keine Daten als Grundlage für Empfehlungen bereitstehen. Streaming-Plattformen wie Netflix versuchen dieses Problem dadurch zu umgehen, dass der Nutzer direkt bei seiner Anmeldung an einer Umfrage teilnehmen muss. Dabei wird er dazu aufgefordert, eine bestimmte Menge an Filmen und Serien, die er bereits gesehen hat, zu bewerten. Eine weitere Möglichkeit dieses Problem zu umgehen ist die Kooperation von verschiedenen Plattformen. Würden beispielsweise Netflix und IMdB (eine Online-Datenbank für Filme und Serien) zusammenarbeiten, könnten bei Netflix Empfehlungen auf Grundlage der Bewertung auf IMdB gemacht werden. Zudem würde die Möglichkeit bestehen, mit einem sozialen Netzwerk zu kooperieren und die Interessen der Freunde als Grundlage für Empfehlungen zu nehmen.
Besonders bei kontextsensitiven Recommender Systemen tritt oft das sparsity-Problem in den Vordergrund. Bei diesem Problem ist die Anzahl der zur Verfügung stehenden Datensätze stark beschränkt. Je mehr Dimensionen kontextueller Filterung verwendet werden, desto eher kommt das sparsity-Problem auf. Sind die Einschränkungen schließlich zu genau, so wird die Anzahl an zutreffenden Daten immer geringer. Dies könnte dazu führen, dass keine sinnvolle Empfehlung für den derzeitigen Kontext ausgesprochen werden kann. [AT11]
Eine weitere Hürde stellt die Transparenz für den Nutzer dar. Hierbei kann zwar zunächst unterschieden werden, ob der Nutzer Erkenntnisse darüber haben soll, nach welchen Kriterien die Empfehlungen ausgesprochen werden sollen, oder nicht. Viele Dimensionen kontextueller Filterung können ohne die manuelle Eingabe durch den Benutzer nur schlecht eingestellt werden. Ein Filteralgorithmus kann beispielsweise automatisch entscheiden, ob es sich um einen Wochentag oder um ein Wochenende handelt, kann jedoch beispielsweise bei Filmen nicht wissen, in welcher Gesellschaft ein Film geschaut wird. Diese Einstellung müsste der Nutzer dann manuell vornehmen.

6 Zusammenfassung

Zunächst wurde erläutert, dass es sich bei Recommender Systemen um Empfehlungssysteme handelt. Diese sind sinnvoll, wenn ein Nutzer sich wegen der Fülle an Daten

nicht mehr zurecht findet. Häufige Anwendungsszenarien sind im eCommerce und bei Streaming-Anbietern. In Kapitel 2 wurden verschiedene Filtermethoden vorgestellt. Die beiden wichtigsten sind inhaltsbasierte und kollaborative Filter. Kollaborative Filter vergleichen Nutzer mit ähnlichen Interessen und stellen auf dieser Grundlage Empfehlungen aus. Sie werden in der Praxis am häufigsten verwendet. Inhaltsbasierte Filter vergleichen Eigenschaften von Items um eine Empfehlung abzugeben. Des Weiteren wurden demographische, nutzenbasierte, wissensbasierte, gruppenorientierte und hybride Recommender Systeme vorgestellt. Hybride Systeme stellen dabei eine Mischform aus verschiedenen Empfehlungssystemen dar.
In Kapitel 3 wurden kontextsensitive Recommender Systeme vorgestellt. Diese präzisieren die Empfehlung durch den Kontext, in dem sich der Benutzer befindet. Dabei stellt jede Kontextebene eine neue Dimension dar und das zweidimensionale Modell von Recommender Systemen wird zu einem Multidimensionalen. Die Brille24 hat die Möglichkeit für ihre Recommender Engine verschiedene Kontextaspekte einzubeziehen. Die wohl sinnvollsten wären Jahreszeiten, demographische Daten und Geo-Daten.
Im vierten Kapitel wird die Recommender Engine der Firma prudsys vorgestellt. Sie befindet sich derzeit bei der Brille24 im Einsatz und bietet verschiedene Konfigurationsmöglichkeiten. Sie kontrolliert sich selbst, indem sie Brillen, die nicht zu den Verkaufsschlagern zählen, mit in die Empfehlungen nimmt um die Auswirkungen der Empfehlungen zu überprüfen.
Im fünften Kapitel werden die Probleme und Grenzen von Recommender Systemen aufgeführt. Das häufigste Problem besteht darin, das für eine Empfehlung zu wenig Daten zur Verfügung stehen. Dieses kann durch aufwendige Machine-Learning Algorithmen umgangen werden.

Literaturverzeichnis

[AT05] Adomavicius, Gediminas; Tuzhilin, Alexander: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE transactions on knowledge and data engineering, 17(6):734–749, 2005.

[AT11] Adomavicius, Gediminas; Tuzhilin, Alexander: Context-aware recommender systems. In: Recommender systems handbook, S. 217–253. Springer, 2011.

[Bu02] Burke, Robin: Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction, 12(4):331–370, 2002.

[HK07] Hohfeld, Stefanie; Kwiatkowski, Melanie: Empfehlungssysteme aus informationswissenschaftlicher Sicht-State of the Art. Information Wissenschaft und Praxis, 58(5):265, 2007.

[MS12] Meier, Andreas; Stormer, Henrik: eBusiness & eCommerce: Management der digitalen Wertschöpfungskette. Springer-Verlag, 2012.

[RRS11] Ricci, Francesco; Rokach, Lior; Shapira, Bracha: Introduction to recommender systems handbook. Springer, 2011.

[SK09] Su, Xiaoyuan; Khoshgoftaar, Taghi M: A survey of collaborative filtering techniques. Advances in artificial intelligence, 2009:4, 2009.

Seminarpase Propose.AI - TensorFlow Framework

Jannis Oeltjen[1]

Abstract: Dieses Dokument enthält die schriftliche Ausarbeitung der Seminarphase für die Projektgruppe Propose.AI, eine zusammengefasste Vorstellung des Machine Learning Frameworks TensorFlow.

Keywords: Machine Learning; TensorFlow; Frameworks

1 Einleitung

TensorFlow ist ein Machine Learning Framework für den Einsatz in großen heterogenen Umgebungen. Es bildet Berechnungen, geteilte Zustände und die Operationen, welche diese Zustände modifizieren in Form eines Datenflussgraphen ab. Die einzelnen Ecken des Graphen können dabei auf mehrere Maschinen und Prozessoren verteilt werden. Unterstützt werden sowohl CPUs und GPUs als auch speziell angepasste ASICs, welche als Tensor Processing Units (TPU) bezeichnet werden [Ab16, S. 265].

Im folgenden Dokument werden die Hintergründe und Ziele vorgestellt, die zur Entwicklung von TensorFlow beigetragen haben. Zudem werden die Design-Prinzipen dargelegt, nach denen TensorFlow entworfen wurde, und die Funktionsweise bzw. das Ausführungsmodell wird erläutert. Den Abschluss bildet eine kurze Vorstellung der Architektur und von einigen Erweiterungen bzw. auf TensorFlow aufbauende Tools.

2 Hintergrund und Geschichte

TensorFlow ersetzte bei Google das bis dahin genutzte System DistBelief, um Schwächen des Vorgängers auszubessern und den Data Scientisten bei Google eine Plattform zu bieten, die leichter erweiterbar ist und aufgrund von einer besseren Skalierbarkeit (in diesem Fall down- und nicht upscaling) flexibler einsetzbar ist [Ab16, S. 266].

DistBelief basiert auf einer sogenannten *parameter server* Architektur. In dieser besteht ein Job aus zwei getrennten Prozesssammlungen; zustandslose *worker* Prozesse, welche

[1] Carl von Ossietzky Universität, Very Large Business Applications, Ammerländer Heerstraße 114, 26129 Oldenburg, Deutschland jannis.oeltjen@uni-oldenburg.de

den Großteil der Berechnungen durchführen, und zustandsbehaftete *parameter server* Prozesse, welche die aktuellen Werte für die Parameter des Modells verwalten und über Maschinengrenzen hinweg synchron halten [Ab16, S. 266].

Eines der Probleme des DistBelief Systems ist die Implementierung neuer Layer. Ein DistBelief Layer ist eine Komposition von mehreren mathematischen Operatoren, wie z.B. Matrixmultiplikation, und anschließender Anwendung einer S-Funktion. Diese Layer sind aus Performancegründen in C++ implementiert. Das hat zur Folge, dass Forscher während der Entwicklung von Modellen sich entweder mit den zur Verfügung stehenden Layern zufrieden geben müssen oder diese selber in einer für sie fremden Sprache implementieren müssen. Das hemmt etwaige Experimente mit neuen Klassifizierern oder anderen Modellen [Ab16, S. 266]. TensorFlow bietet genau wie DistBelief eine Schnittstelle in Python an, erlaubt aber neben der Nutzung der vorgefertigten API und der bestehenden Layer auch die Entwicklung eigener komplexer Modelle und Algorithmen. Auch TensorFlow verwendet in C++ geschriebene Komponenten um Performancegewinne durch neue Befehlssatzerweiterungen in modernen Prozessoren zu nutzen. Anders als bei DistBelief werden jedoch nicht vollständige Layer [De12, S. 3] implementiert, sondern feingranulare einzelne Operatoren und einfache Algorithmen. Dadurch hat der Nutzer die Möglichkeit, diese nach seinem Gusto neu zu ordnen, ohne dass ein Sprachwechsel zwischen Python und C++ stattfinden muss.

Ein weiterer Kritikpunkt an DistBelief ist die Schwierigkeit der Verbesserung von Trainingsalgorithmen. In vielen neuronalen Netzwerken wird das sogenannte stochastic gradient descent (SDG) angewandt. Bei diesem Vorgang werden die Parameter des neuronalen Netzwerkes iterativ verbessert, indem sie in die Richtung verändert werden, welche maximal das Ergebnis der Loss-Funktion verringert. Bei der Optimierung von Trainingsalgorithmen wird i.d.R die Update-Funktion geändert, was bei DistBelief jedoch nur durch eine Änderungen der *parameter server* Implementierung möglich ist. Zusätzlich erschwerend kommt hinzu, dass die *put()* und *get()* Interfaces des *parameter servers* nicht ideal gewählt sind, da es für manche Modelle wichtig wäre, mehrere Parameter in einem atomaren Vorgang zu aktualisieren. [Ab16, S. 266]

Letztlich ist auch die Entwicklung von neuen Trainingsalgorithmen für DistBelief dadurch eingeschränkt, dass die Worker in dem System einem fixen Ausführungspattern folgen. Der Ablauf folgt immer diesem Pattern:

1. Einlesen eines Datensatzes und der aktuellen Parameterwerte für das Modell
2. Berechnung der Loss-Funktion (Forwad pass durch das Netzwerk)
3. Verläufe für die Parameter berechnen (Backward pass)
4. Parameterverläufe an den *parameter server* senden

Dieses Pattern eignet sich gut für einfache Feed-Forward Neuronale Netze ohne Feedbackschleifen oder Zyklen, schlägt aber fehl bei komplexeren Netzwerken, wie z.B. Recurrent

Neural Networks (RNNs) oder Convolutional Neural Networks (CNNs). DistBelief kann hier nur einen kleinen Teilbereich der Datenanlalyse durchführen. TensorFlow dagegen bietet die Möglichkeit, für verschiedenste Problemstellungen und Algorithmen eine geeignete Laufzeitumgebung zur Verfügung zu stellen [Ab16, S. 266].

Bei der Entwicklung von DistBelief wurde nur eine Umgebung berücksichtigt; der Betrieb auf großen verteilten Mehrkernsystemen. Auch wenn es mittlerweile möglich ist, GPUs für das Trainieren von Modellen mittels DistBelief zu nutzen, fehlt dem Programm eine entscheidende Eigenschaft. Es skaliert nicht nach unten. Dies mag ungewöhnlich klingen, aber die Fähigkeit von TensorFlow, auf einzelnen Workstations von Mitarbeitern und Forschern genutzt zu werden, stellt einen massiven Gewinn gegenüber DistBelief dar. Durch die Möglichkeit, ein Modell im kleinen Maßstab auszuprobieren, kann die Entwicklung von diesen wesentlich flexibler gestaltet werden. Die lokal getesteten Modelle und der Code für deren Erstellung können dann mit nur wenig Aufwand in den produktiven Einsatz gebracht werden [Ab16, S. 267]. TensorFlow bietet damit ein einheitliches Programmiermodell und eine Laufzeitumgebung für verschiedenste Umgebungen, von einzelnen Workstations bis hin zu großen verteilten GPU gestützten Clustern [Ab16, S. 267].

3 Design Prinzipien

Wie bereits im vorherigen Abschnitt erläutert, war eines der Entwicklungsziele für TensorFlow eine im Vergleich zu DistBelief höhere Flexibilität und bessere Skalierbarkeit in beide Richtungen. Dies wurde unter anderem durch die Einhaltung von drei Kern-Designprinzipien erreicht. Diese werden im Folgenden erläutert.

3.1 Datenflussgraphen aus primitiven Operatoren

Sowohl DistBelief als auch TensorFlow nutzen Datenflussgraphen zur Repräsentation ihrer Modelle, der Unterschied ist dabei, dass DistBelief Modelle i.d.R. aus wenigen komplexen Ebenen besteht, während bei TensorFlow die einzelnen verwendeten mathematischen Operatoren als einzelne Elemente im Graphen abgebildet werden [Ab16, S. 267]. Ein Element im Graph kann dann z.B. eine Matrixmultiplikation (*MatMul*) sein.

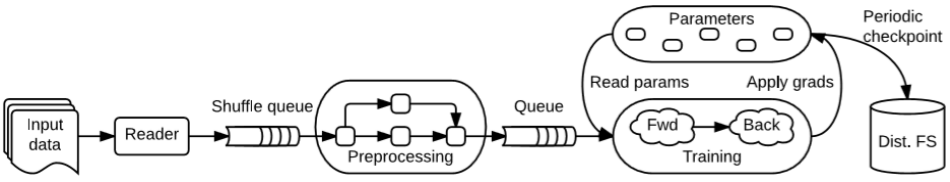


Abb. 1: Schematischer Datenflussgraph für das Training eines Modells [Ab16, S. 269]

Die Abbildung 1 zeigt schematisch einen solchen Datenflussgraphen inklusive Subgraphen für das Einlesen sowie Preprocessing von Daten, Training und dem Erstellen von Checkpoints.

Dies schafft einen hohes Maß an Flexibilität, da anders als bei DistBelief nicht in C++ geschriebene Layer für Anpassungen des Modells verändert werden müssen, sondern einzelne Elemente des Graphen können ausgetauscht oder angepasst werden. Dadurch haben Nutzer die Möglichkeit, neue Algorithmen zu erstellen, ohne die High-Level-API von TensorFlow zu verlassen [Ab16, S. 267].

3.2 Verzögerte Ausführung

Das zweite Design-Prinzip, welches bei der Entwicklung von TensorFlow berücksichtigt wurde, ist die sogenannte verzögerte Ausführung. Dieses Prinzip besagt, dass die eigentliche Ausführung eines Modells getrennt von den vorbereitenden Schritten abläuft. Bei TensorFlow äußert sich dies darin, dass eine typische Anwendung aus zwei unabhängigen Phasen besteht. Die erste Phase beinhaltet die Definition des Programms, z.B. ein neuronales Netz und die dazugehörigen Update-Funktionen. Daraus wird ein symbolischer Datenflussgraph aufgebaut, welcher Platzhalter für eingehende Daten sowie Variablen für den Zustand des Netzes enthält. Die zweite Phase ist die Ausführung des in Phase eins entstandenen Graphen auf den verfügbaren Ressourcen/ Maschinen [Ab16, S. 268]. Durch die Verzögerung der Ausführung des Graphen bis zu dem Zeitpunkt, zu dem das Programm vollständig geladen ist, kann TensorFlow die Ausführung mithilfe von globalen Informationen über die Berechnung optimieren. Diese Designentscheidung macht die Ausführung von Graphen wesentlich effizienter, bewirkt gleichzeitig aber auch, dass komplexere Funktionen für den Ausführungsgraphen, insbesondere Feedbackfunktionen, in dem Datenflussgraphen selber abgebildet werden müssen. Nähere Informationen dazu im Abschnitt 4.4.

3.3 Gemeinsame Abstrahierungsebene für heterogene Beschleuniger

Maschine Learning Modelle können auf CPUs, GPUs oder auch speziell angefertigter Hardware wie den sogenannten TPUs (Tensor Processing Units) trainiert werden. Das Design-Prinzip der gemeinsamen Abstrahierung für heterogene Beschleuniger gibt vor, dass es für die von der High-Level-API genutzten Interfaces Implementierungen für verschiedene Plattformen gibt, so dass ein und derselbe Graph auf mehreren Plattformen lauffähig ist und so z.B. die massiven Performancegewinne durch TPUs im Vergleich zu CPUs genutzt werden können, ohne dass Aufwände für die Anpassung der Anwendung entstehen [Ab16, S. 268]. Damit ein angepasster Chip, TPU, FPGA oder ASIC für TensorFlow genutzt werden können, müssen mindestens folgende drei Funktionen verfügbar sein:

1. Bereitstellung eines Kernels zur Ausführung
2. Allokation von Arbeitsspeicher für Ein- und Ausgaben
3. Transfer von Buffern zu und vom Arbeitsspeicher

Jeder mathematische Operator, wie z.B. Matrixmultiplikation, kann mehrere gerätespezifische Implementierungen haben. Der Austausch von Informationen zwischen den Elementen des Datenflussgraphen geschieht auf Basis eines gemeinsamen Austauschformates, welches ausschließlich aus primitiven Datentypen besteht.

Dies hat noch weitere Auswirkungen auf die Funktionsweise von TensorFlow. Eine der Hauptkonsequenzen ist, dass es so etwas wie einen *parameter server* wie in DistBelief nicht gibt, da TensorFlow in einem Cluster als eine Ansammlung von *Tasks* bereitgestellt wird. Ein Task ist dabei ein Prozess, der mit anderen Prozessen Informationen austauschen kann. Jeder Task stellt dabei die gleiche Graph-Ausführungs-API bereit und enthält eines oder mehrere Geräte [Ch15, S. 4]. Typischerweise nimmt ein Teil dieser Tasks dabei eine Rolle ein, welche ungefähr dem *parameter server* in anderen Systemen entspricht. Diese werden *PS tasks* genannt, die anderen *worker tasks*. Diese *PS tasks* können beliebige TensorFlow Graphen ausführen und sind dadurch flexibler als konventionelle *parameter server*. Nutzer können diese mit dem gleichen Interface programmieren, mit dem auch die Modelle definiert werden [Ab16, S. 268].

4 Ausführungsmodell

TensorFlow bildet Programme, wie bereits hinreichend erläutert, in Datenflussgraphen ab. Ein Programm wird dabei durch genau einen Ausführungsgraphen mit wechselnder Komplexität repräsentiert. Der Graph enthält alle Berechnungen und Stati des Machine Learning Algorithmus einschließlich der individuellen mathematischen Operationen, der Parameter mit ihren jeweiligen Update-Funktionen und der Eingabeverarbeitung (siehe 1). Im Folgenden werden die Eigenschaften der Bestandteile des Graphen erläutert:

4.1 Elemente im Datenflussgraphen

Jeder Eckpunkt im Graph repräsentiert eine lokale Berechnung und jede Kante die Ein- oder Ausgabe von/ zu einem Eckpunkt. Die Berechnungen in Eckpunkten werden als Operationen bezeichnet, die Werte, auf denen die Berechnungen durchgeführt werden, Tensoren. Im Folgenden werden die Unterschiede zwischen Tensoren und Operationen erläutert, außerdem die beiden Sonderfälle Variablen und Queues erklärt, bei denen es sich in diesem Kontext um zustandsbehaftete Operationen handelt [Ab16, S. 269f].

Tensoren In TensorFlow werden alle Daten als n-dimensionale Arrays, sogenannte Tensoren modelliert. Die Elemente haben dabei einen von wenigen unterschiedlichen primitiven Datentypen wie *int32*, *float32* oder *string*, wobei *string* auch beliebige binäre Daten repräsentieren kann [Ab16, S. 270]. Tensoren bilden dabei die natürlichen Eingabeparameter für viele verschiedene mathematische Operationen, die im Maschinellen Lernen eingesetzt

werden. Eine Matrixmultiplikation z.B. würde zwei zweidimensionale Tensoren als Eingabe erwarten und einen zweidimensionalen Tensor als Ausgabe erzeugen. Auf unterster Ebene sind Tensoren in TensorFlow dicht, d.h. sie enthalten keine Null-Werte. *Sparse data*, Daten mit Null-Werten, können jedoch trotzdem abgebildet werden; entweder durch unterschiedlich lange *string*-Elemente in einem dichten Tensor oder durch die Nutzung von mehreren Tensoren, wobei einer dabei die Koordinaten der vorhandenen Daten in dem anderen Tensor angibt [Ab16].

Operationen Wie bereits im einleitenden Text für den Abschnitt erläutert, handelt es sich bei Operationen i.d.R. um mathematische Operatoren. Per Definition erwartet eine Operation $m \geq 0$ Tensoren als Eingabe und produziert $n \geq 0$ Tensoren als Ausgabe. Dabei hat jede Operation einen Typ, wie z.B. *Const*, *MatMul* oder *Assign*. Neben diesen Typ können noch optionale Attribute vorhanden sein, welche das Verhalten der Operation zum Kompilierzeitpunkt (während des Aufbaus des Datenflussgraphens), das Verhalten der Operation verändern. Die einfachste Operation *Const*, aufgerufen durch `tf.constant()`, hat keinen Eingabetensor und erzeugt einen Tensor als Ausgabe. Für den Wert der Konstante existiert ein Attribut, welches den Rückgabewert der Operation zum Kompilierzeitpunkt festlegt [Ab16, S. 270].

Variablen Operationen können einen Zustand halten, der sich bei jeder Ausführung verändert. Dafür werden sogenannte *Variable* Operationen genutzt. Diesen ist ein veränderbarer Buffer zugeordnet, welcher zum Teilen von Parametern des Modells genutzt werden kann. Eine *Variable* hat keine Eingabe und erzeugt anders als die normalen Operationen keinen Tensor als Ausgabe, sondern ein sogenanntes *reference handle*, welches den typisierten Lese- und Schreibzugriff auf den Buffer erlaubt. Um Daten aus einer Variable zu lesen, wird die *Read* Operation genutzt, welche ein *reference handle* als Eingabe erwartet und einen dichten Tensor mit dem Wert der Variable zurück gibt. Neben der *Read* Operation existieren auch Operationen, welche den Zustand der Variable ändern können. Ein Beispiel dafür ist die simpelste Form der Addition durch *AssignAdd*, welche wieder ein *reference handle r* und einen Tensor x erwartet und zum Ausführungszeitpunkt dieses Update ausführt $State'[r] \leftarrow State[r] + x$ [Ab16, S. 270].

Queues Neben Variablen unterstützt TensorFlow auch mehrere Formen von Queues. Diese können zur fortgeschrittenen Koordination von Eingabedaten genutzt werden. Die einfachste unterstützte Queue ist die *FIFOQueue*, welche intern eine Queue von Tensoren hält und gleichzeitigen Zugriff auf diese erlaubt. Die Funktionsweise ist ähnlich wie bei *Variable* Operationen. Es wird ein *reference handle* zurückgegeben, mit dem Standard Queueoperationen wie *Enqueue* und *Dequeue* ausgeführt werden können. Ein Einsatzbereich von Queues in TensorFlow ist das Preprocessing von Eingabedaten. *Dequeue* und *Enqueue* blockieren jeweils, wenn die Queue leer bzw. voll ist. Dies erzeugt Backpressure im

Preprocessing und verhindert eine Überlastung der Computing Nodes im Cluster. In Zusammenhang mit dem dynamischen Kontrollfluss (siehe Abschnitt 4.4) kann so eine Art von Stream Processing zwischen Subgraphen implementiert werden [Ab16, S. 270].

4.2 Aufgeteilte und gleichzeitige Ausführung

Die API zur Ausführung eines Graphen erlaubt jedem Client deklarativ zu beschreiben, welcher Subgraph ausgeführt werden soll. Jeder Aufruf der API zur Ausführung wird *step* genannt. TensorFlow unterstützt die Ausführung von mehreren *steps* gleichzeitig, diese werden als *concurrent steps* bezeichnet. Zustandsbehaftete Operationen wie *Variable* oder *FIFOQueue* erlauben den Austausch von Informationen, oder wenn nötig, auch die Synchronisation von gleichzeitig ausgeführten Subgraphen [Ab16, S. 270f]. Abbildung 1 zeigt eine typische Trainingsanwendung, bei der mehrere Subgraphen gleichzeitig ausgeführt werden; der Trainingssubgraph, das Preprocessing sowie der Checkpoint Graph.

4.3 Verteilte Ausführung

Aufbauend auf die Fähigkeit zur aufgeteilten und gleichzeitigen Ausführung kommt eine der größten Stärken von TensorFlow hinzu; die verteilte Ausführung auf einer Vielzahl von heterogenen Geräten wie CPUs, GPUs oder TPUs. Geräte sind wiederum einem *Task* (siehe 3.3) zugeordnet. Die TensorFlow Runtime platziert unter Berücksichtigung von impliziten und expliziten Vorgaben die einzelnen Operationen des Gesamtgraphen auf verschiedenen Geräten. Die Zuweisungsrichtlinien von *Tasks* wie dem *worker task* oder dem *PS task* auf die verfügbaren Geräte können von erfahrenen Nutzern angepasst werden, um die zur Verfügung stehenden Ressourcen optimal auszunutzen [Ab16, S. 271f]. Sobald alle Operationen verteilt wurden und der Subgraph für die *steps* berechnet wurde, werden die Operationen von TensorFlow nochmals in gerätebezogene Subgraphen aufgeteilt. Diese enthalten jeweils neben allen Operationen die dem jeweiligen Gerät zugeordnet sind, noch weitere *Send* und *Recv* Operationen, welche Eckpunkte im Datenflussgraphen ersetzen, die über Gerätegrenzen hinweg gehen würden [Ab16, S. 271].

4.4 Dynamischer Kontrollfluss

Fortgeschrittene Machine Learning Algorithmen wie z.B. Recurrent Neural Networks (RNN) erfordern die Fähigkeit der Anpassung des Graphen zur Laufzeit. TensorFlow nutzt jedoch das Prinzip der Verzögerten Ausführung, bei welchem der vollständige Datenflussgraph vor der Ausführung berechnet wird (siehe Abschnitt 3.2). Um den Datenfluss zur Laufzeit basierend auf den Eingabedaten zu steuern, werden Bedingungen und Schleifen in den Graphen selber eingefügt. Genutzt werden dazu die beiden Operationen *Switch* und *Merge*.

Switch ist dabei ein Demultiplexer, welcher einen Dateneingang, einen Kontrolleingang und zwei Ausgänge hat. Basierend auf dem Kontrolleingang wird einer der beiden zur Verfügung stehenden Ausgänge gewählt [AC86]. *Merge* ist die Gegenoperation zu *Switch* und führt zwei Eingänge zu einem zusammen [AC86]. Diese Mittel werden von TensorFlow genutzt, um sogar überlappende parallele Ausführungen von Iterationen des Modells zu ermöglichen. Diese können dann auch wieder mit den TensorFlow Boardmitteln auf mehrere Geräte verteilt werden [Ab16, S. 272].

5 Architektur / Implementierung

TensorFlow nutzt, wie viele, moderne Anwendungen, eine Schichtenarchitektur (siehe Abbildung 2), welche Cross-Plattform-Kompatibilität bei gleichzeitiger Reduktion der Komplexität für den Enduser ermöglicht [Ab16, S. 275]. Die Trennebene in der Architektur ist eine C-API, welche den Benutzercode von der Kernanwendung abkapselt. TensorFlow selber ist im Kern in C++ geschrieben. Die Auswahl ist auf diese Sprache gefallen, um maximale Portierbarkeit zu gewährleisten. So ist diese neben Linux, OS X und Windows auf auch Android, iOS lauffähig und ist dabei kompatibel zu unterschiedlichen CPU Architekturen wie x86 und verschiedenen ARM Varianten. Zusätzlich werden auch GPU Microarchitekturen wie NVIDIAs Pascal, Kepler und Volta GPUs unterstützt [Ab16, S. 275]. Unter der C-API gibt es zwei Implementierungen, den *distributed master*, der, wie der Name suggeriert, den Master Prozess darstellt und Benutzereingaben in *tasks* wandelt. Diese werden dann von *dataflow executors* bearbeitet. Aufgabenverteilung und Ausführung sind explizit getrennt, was u.a. eine Bedingung für die Fähigkeit zur verzögerten Ausführung (siehe Abschnitt 3.2) ist.

In Abbildung 2 sind darunter die Kernel für verschiedene Operationen wie z.B. die bereits erläuterte *Const* oder *Queue* Operation angeordnet (siehe Abschnitte 4.1 und 4.1).

Neben den Kernels für die Operationen gibt es noch die beiden Schichten Netzwerk und Gerät. In der Netzwerkschicht befindet sich z.B. der Code für die Remote Procedure Calls (RPC) und dem Remote direct memory access (RDMA), welcher insbesondere für die verteilte Ausführung von TensorFlow Anwendungen von Priorität ist. In der Geräte-Schicht befindet sich der spezifische Code für die unterschiedlichen Geräte, die im Zusammenhang mit TensorFlow verwendet werden können wie z.B. CPUs, GPUs und TPUs [Ab16, S.m 275f].

Nutzer arbeiten ausschließlich in den Schichten, die in der Architektur über der C-API angeordnet sind (siehe Abbildung 2). Es existieren Client-Bibliotheken für verschiedene Sprachen wie Python, C++ und Java, welche TensorFlow über die C-API ansprechen und die Schnittstelle zwischen Nutzer und TensorFlow darstellen. Neben den Client-Bibliotheken existieren wiederum eine Vielzahl von Bibliotheken, die im Zusammenhang mit TensorFlow genutzt werden, welche selber auch die Client-Bibliotheken von TensorFlow nutzen. Beispiele dafür sind z.B. Keras, eine High-Level Python API für die Entwicklung

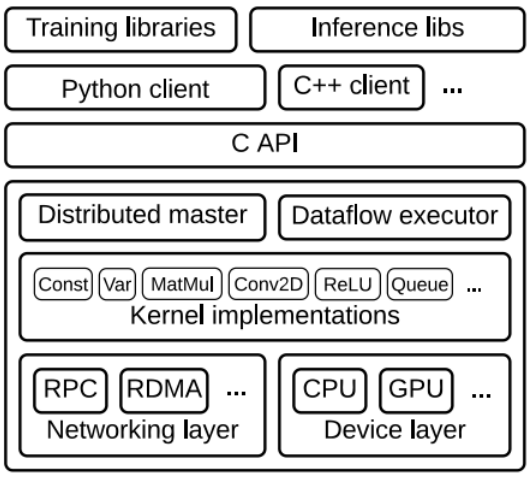


Abb. 2: Schichten der TensorFlow Architektur [Ab16, S. 275]

von Neuronalen Netzen, welche auf Frameworks wie TensorFlow, Theano oder CNTK aufsetzt [Ke18] und TensorFlowOnSpark, eine Schnittstelle von TensorFlow zu Apache Spark, um Deep Learning in Apache Hadoop Clustern zu ermöglichen [Ya17].

6 Erweiterungen

Durch die Designentscheidung bei TensorFlow, alle Funktionalitäten über APIs anzusprechen, wurde eine profunde Grundlage für die Entwicklung von Erweiterungen geschaffen. Im folgenden Abschnitt werden repräsentativ die beiden Entwicklungen TensorBoard und die JavaScript Implementierung von TensorFlow, TensorFlow.js, vorgestellt.

6.1 TensorBoard

TensorBoard ist eine grafische Oberfläche für TensorFlow. Sie enthält eine Sammlung von Tools, um Graphen zu zeichnen, quantitative Metriken abzubilden und zusätzliche arbiträre Daten anzuzeigen [Go18]. Abbildung 3 zeigt einen Screenshot einer voll konfigurierten TensorBoard Instanz.

TensorBoard kann mithilfe von sogenannten *summary ops* Daten von TensorFlow empfangen. Dies sind Operationen, wie z.B. *MatMul*, die bereits in diesem Dokument erläutert wurden. Sie nehmen einen oder mehrere Tensoren als Eingabe und produzieren solche als Ausgabe. Der Unterschied zu den TensorFlow Operationen ist jedoch, dass die Ausgabetensoren keine primitiven Daten enthalten, sondern mit Google Protocol Buffers serialisierte Daten, welche

auf ein Speichermedium geschrieben und anschließend an TensorBoard gesendet werden [Te18a].

Es stehen dabei die *summary ops* Operationen *Scalar*, *Image*, *Audio*, *Text* und *Histogram* zur Verfügung [Te18a]. Für jeden Datentyp gibt es geeignete Visualisierungen, wie z.B. das *Scalar Dashboard* oder das *Distribution Dashboard* (zur Darstellung von Verteilungen in Histogrammen).

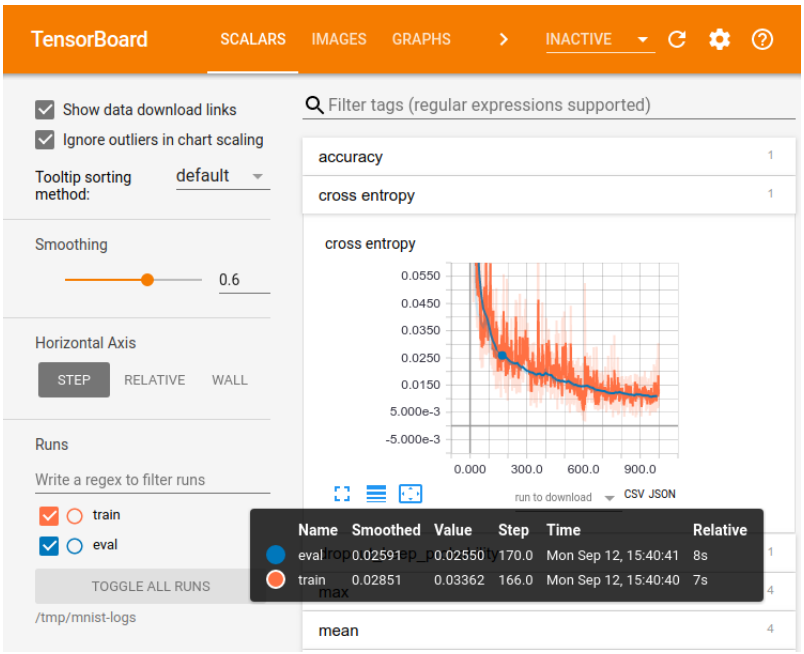


Abb. 3: TensorBoard Screenshot [Go18]

6.2 TensorFlow.js

TensorFlow.js ist eine vollständig in JavaScript und WebGL geschriebene alternative Implementierung der TensorFlow API. Sie erlaubt es, Maschine Learning Modelle im Browser zu entwickeln, viel wichtiger aber erlaubt sie auch das Laden und Ausführen von Modellen, die mit TensorFlow erstellt wurden. Dadurch kann ein Modell auf einem großen leistungsstarken System initial trainiert werden, um dann auf leistungsschwachen mobilen Endgeräten ausgeführt zu werden. Außerdem ist es möglich, ein bestehendes Modell weiter zu trainieren z.B. mit Sensordaten des mobilen Endgeräts. TensorFlow.js ist dabei nicht auf den Import von TensorFlow Modellen beschränkt, sondern kann auch Keras Modelle importieren [Te18b]. Dadurch eignet es sich gut für mobile Anwendungen, die plattformunabhängig als Web-Anwendung entwickelt werden, insbesondere Apacha

Cordova oder PhoneGap basierende Apps können so ohne nativen Code von Machine Learning Algorithmen profitieren.

TensorFlow.js ist aus dem Projekt deeplearn.js entstanden, welches im März 2018 in das TensorFlow Projekt überführt und in dem Zuge zu TensorFlow.js umbenannt wurde [Th18].

7 Fazit

TensorFlow biete eine mächtige Machine Learning Plattform, die sich durch eine ausgezeichnete Skalierungsfähigkeiten sowohl nach oben als auch nach unten auszeichnet. Der modulare Ansatz und die strikte Trennung von API und Implementierung schaffen ein hohes Maß an Flexibilität sowohl in der Auswahl der genutzten Tools als auch bei der Zielplattform. Die Betreuung des Projektes durch finanzstarke Unternehmen wie Google schaffen ein gewisses Maß an Sicherheit, das insbesondere bei der Auswahl von Frameworks ein nicht zu vernachlässigender Aspekt ist.

Durch das hohe Maß an Flexibilität und Skalierbarkeit hat sich um TensorFlow herum bereits ein Ökosystem von aufbauenden Bibliotheken und Tools gebildet, die wiederum den Mehrwert von TensorFlow erheblich steigern. So ist es nicht verwunderlich, dass TensorFlow zu den populärsten Machine Learning Frameworks zur Zeit gehört [Uj18].

Literatur

[Ab16] Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; Kudlur, M.; Levenberg, J.; Monga, R.; Moore, S.; Murray, D. G.; Steiner, B.; Tucker, P.; Vasudevan, V.; Warden, P.; Wicke, M.; Yu, Y.; Zheng, X.: TensorFlow: A System for Large-Scale Machine Learning. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). USENIX Association, Savannah, GA, S. 265–283, 2016, ISBN: 978-1-931971-33-1, URL: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi.

[AC86] Arvind; Culler, D. E.: Dataflow Architectures. Annual Review of Computer Science 1/, hrsg. von Annual Reviews Inc., S. 225–253, 1986, URL: www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&%20doc=GetTRDoc.pdf&AD=ADA166235, Stand: 06. 05. 2018.

[Ch15] Chen, T.; Li, M.; Li, Y.; Lin, M.; Wang, N.; Wang, M.; Xiao, T.; Xu, B.; Zhang, C.; Zhang, Z.: MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. CoRR abs/1512.01274/, 2015, arXiv: 1512.01274, URL: http://arxiv.org/abs/1512.01274.

[De12] Dean, J.; Corrado, G.; Monga, R.; Chen, K.; Devin, M.; Mao, M.; Ranzato, M.; Senior, A.; Tucker, P.; Yang, K.; Le, Q. V.; Ng, A. Y.: Large Scale Distributed Deep Networks. In (Pereira, F.; Burges, C. J. C.; Bottou, L.; Weinberger, K. Q., Hrsg.): Advances in Neural Information Processing Systems 25. Curran Associates, Inc., S. 1223–1231, 2012, URL: http://papers.nips.cc/paper/4687-large-scale-distributed-deep-networks.pdf.

[Go18] Google: TensorBoard: Visualizing Learning, 28. Apr. 2018, URL: https://www.tensorflow.org/programmers_guide/summaries_and_tensorboard, Stand: 07. 05. 2018.

[Ke18] Keras Team: Keras: Deep Learning for humans, 6. März 2018, URL: https://github.com/keras-team/keras/blob/master/README.md, Stand: 07. 05. 2018.

[Te18a] TensorFlow Project: TensorBoard, 3. Mai 2018, URL: https://github.com/tensorflow/tensorboard/blob/master/README.md, Stand: 07. 05. 2018.

[Te18b] TensorFlow Team: TensorFlow.js, 3. Mai 2018, URL: https://github.com/tensorflow/tfjs/blob/master/README.md, Stand: 07. 05. 2018.

[Th18] Thorat, N.: TensorFlow.js Core API, 30. März 2018, URL: https://github.com/tensorflow/tfjs-core/releases/tag/v0.6.0, Stand: 07. 05. 2018.

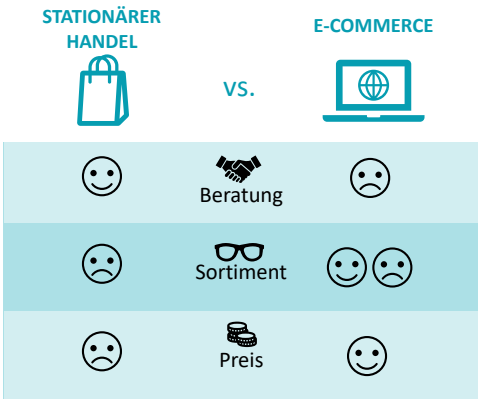
[Uj18] Uj, A.: Five Most Popular Open Source Frameworks Used in Machine Learning, hrsg. von Analytics Insight, Stravium Intelligence LLP, 14. Apr. 2018, URL: https://www.analyticsinsight.net/five-most-popular-open-source-frameworks-used-in-machine-learning/.

[Ya17] Yahoo: TensorFlowOnSpark, 15. Juni 2017, URL: https://github.com/yahoo/TensorFlowOnSpark/blob/master/README.md, Stand: 07. 05. 2018.

A.9 Handouts der Teilprojekte

PROPOSE.AI - Brillenberater

Problemstellung & Ziel



Die zentrale Problemstellung liegt darin, dass die persönliche Beratung beim Brillenkauf, wie sie im stationären Handel gegeben ist, bisher online nicht im gleichen Maß umgesetzt wird. Ziel ist es, die Intuition und Erfahrung von Optikern in den Online-Handel zu übertragen, um dem Kunden so aus dem großen Sortiment eine personalisierte Empfehlung geben zu können.

KI als Werkzeug

Um das oben genannte Ziel zu erreichen, kommt künstliche Intelligenz zum Einsatz. Das bedeutet in diesem Fall, der Maschine beizubringen, das zu machen, was bei der lokalen Beratung in Sekunden passiert.

Notwendige Schritte

1. Auswählen der entscheidenden Merkmale für die Empfehlung von Brillen
2. vorhandene Datensets sichten, aufbereiten und bereinigen
3. ggf. eigene Datensets aufbauen, d. h. Bilder mit Label versehen
4. künstliche neuronale Netze für die Erkennung der Merkmale entwerfen
5. Netze mit der zusammengestellten Datengrundlage trainieren
6. Ergebnisse auswerten
7. ggf. Anpassung der Netze
8. Regeln für die Brillenempfehlung erstellen
9. Mapping von Brillen und Merkmalen

Der Weg zur Brillenempfehlung

Der Brillenberater stellt den Kunden in den Vordergrund, indem ihm eine optimale, individuelle Beratung geboten wird. Anhand eines aufgenommenen Portraitfotos werden Gesichtsmerkmale des Kunden mittels künstlicher Intelligenz extrahiert. Durch spezifische Regeln, welche die Intuition eines Optikers im stationären Handel simulieren, werden die passenden Brillen zu den Merkmalen gefunden. Hierdurch wird eine Auswahl von Brillen herausgefiltert, die optimal dem Typen des Kunden entsprechen.

Was steckt dahinter?

Es wurden **7** Datensets mit insgesamt über **270.000** Bildern genutzt, um folgende Merkmale mit künstlichen neuronalen Netzen aus einem Gesicht extrahieren zu können:

- Haar- und Hautfarbe,
- Alter und Geschlecht,
- Gesichtsbehaarung und
- Gesichtsform.

Die entstandenen Netze erzielen durch das Training mit den Daten und weiterem Feintuning eine durchschnittliche Genauigkeit von **95%**.

Parallel dazu wurde eine Auswahl an Brillen aus dem Brille24-Sortiment getroffen und deren Eigenschaften präzise beschrieben.

Mit dieser Grundlage wurden Regeln erstellt, für welche Merkmalsausprägung welche Brillen-eigenschaften geeignet sind und ein anschließendes Mapping vorgenommen.

Ein ovales Gesicht gilt als Brillengesicht und kann jede Brillenform tragen.

So funktioniert's

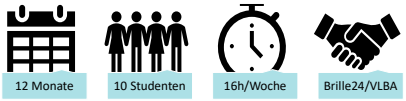
1. Bild wird in den Brillenberater geladen
2. Face Detection und Cropping

3. zugeschnittenes Gesichtsbild durchläuft die künstlichen neuronalen Netze
4. Merkmale werden extrahiert und gesammelt
5. Mapping von erkannten Merkmalen und Brillensortiment unter Anwendung der Regeln
6. Brillenvorschläge werden ausgegeben

Hier geht's zum Brillenberater!
propose-ai.joelt.de

Über PROPOSE.AI

PROPOSE.AI ist eine Projektgruppe der Carl von Ossietzky Universität Oldenburg. Betreut wird das Projekt von der Abteilung Wirtschaftsinformatik / Very Large Business Applications (VLBA) unter der Leitung von Prof. Dr.-Ing. Jorge Marx Gómez. PROPOSE.AI läuft seit April 2018 in Kooperation mit dem Online-Optiker Brille24.



Projektziel

Entwicklung einer AI Service Plattform für …

- Produktempfehlungen für Kunden,
- die Erleichterung des Einkaufsprozesses für Kunden,
- die Optimierung des Beschaffungsprozesses für Brille24 und
- die Optimierung des Marketingprozesses für Brille24

… auf Basis von Bild-, Shop- und Social Media-Daten.

Weitere Teilprojekte

Brillenpassleser

Der Brillenpassleser soll die Customer Experience des Kunden verbessern, indem ihm Arbeit abgenommen wird. Dabei kann der Kunde ein Foto seines Brillenpasses aufnehmen und hochladen. Die Werte werden dann durch den Brillenpassleser ausgelesen und automatisch in das Kundenkonto übertragen.

Trendanalyse

Der Fokus liegt bei diesem Projekt nicht auf dem Kunden, sondern auf den Mitarbeitern von Brille24. Diese sollen bei Entscheidungen in Einkaufs- und Marketingprozessen unterstützt werden. Dazu werden anhand von Bilddaten Trends erkannt und prognostiziert. Mittels künstlicher Intelligenz werden aus diesen Bildern Informationen zu Brillentyp, Rahmenfarbe und weiteren Eigenschaften extrahiert und mit zusätzlichen Daten verknüpft. Diese werden dann in einem Dashboard visualisiert.

Brillentinder

Mit dem Brillentinder soll der Kunde ebenfalls bei der Suche nach einer Brille unterstützt werden. Anders als beim Brillenberater liegt hierbei der Fokus auf dem persönlichen Geschmack des Kunden. Der Kunde kann ihm angezeigte Brillen bewerten und nähert sich so seinen präferierten Brillenmodellen.

PROPOSE.AI - Brillenpassleser

CARL VON OSSIETZKY universität

VERY LARGE BUSINESS APPLICATIONS

BRILLE24 Research

Problemstellung & Ziel

Um Online eine Brille kaufen zu können, muss der Kunde seine Sehwerte übermitteln. Dazu muss er seinen Brillenpass zur Hand nehmen und diese händisch in die Maske übertragen.

Ziel des Brillenpasslesers ist es nun, ihm diesen Arbeitsschritt abzunehmen. Der Kunde soll nur noch ein Foto von seinem Brillenpass machen und die Sehwerte werden automatisch übertragen.

Besonders auf mobilen Endgeräten soll so die Customer Experience verbessert werden. Der Kunde spart Zeit und es werden mögliche Fehlerquellen reduziert.

KI als Werkzeug

Um das oben genannte Ziel zu erreichen, kommt künstliche Intelligenz zum Einsatz. Hierzu wird das vortrainierte Netz von Tesseract verwendet. Dieses ist in der Lage Textzeichen zu erkennen. Zudem wurde ein Netz darauf trainiert die Eckpunkte der Brillenpässe zu finden.

Notwendige Schritte

1. Kunde nimmt ein Bild von sich auf und bestätigt dieses
2. Bild wird hochgeladen
3. Eckpunkte des Passes werden automatisch erkannt
4. perspektivische Transformation setzt das Bild in die Vogelperspektive
5. Bild wird auf eine Einheitsgröße skaliert
6. Herausgeber des Passes wird klassifiziert
7. um die einzelnen Textelemente werden Bounding Boxes gezogen
8. Text in den Bounding Boxes wird über Tesseract ausgelesen
9. erkannter Text wird zusammen mit den Koordinaten des Textes zurückgegeben

Einlesen der Brillenwerte

Der Brillenpassleser soll die Customer Experience des Kunden verbessern, indem ihm das Eintippen der Brillenwerte abgenommen wird. Dazu kann der Kunde ein Foto seines Brillenpasses aufnehmen und hochladen. Die Werte werden dann durch den Brillenpassleser ausgelesen und automatisch in das Kundenkonto übertragen.

So funktioniert's

Texterkennung
Tesseract ist eine Open Source Lösung zur Texterkennung. Dabei zerlegt Tesseract den Text in Textblöcke und liest diese aus. In den ersten Versionen arbeitete Tesseract mit Mustervergleichen. Seit Ende 2016 stellt Tesseract auch eine Texterkennung durch neuronale Netze zur Verfügung. Dadurch konnte die Erkennungsrate spürbar verbessert werden.

Perspektivische Transformation
Durch die perspektivische Transformation wird das Bild von dem Brillenpass automatisch ausgerichtet. Wird bei der Aufnahme des Brillenpasses aus einem zu spitzen Winkel fotografiert, wird es später schwieriger die Koordinaten des erkannten Textes zuzuordenen. Anhand der erkannten Eckpunkte des Brillenpasses wird das Bild perspektivisch transformiert und in die Vogelperspektive gebracht.

Klassifizierung des Herausgebers
Zur Klassifizierung des Brillenpass-Herausgebers wurden verschiedene Ansätze getestet. Ein Histogrammvergleich sollte das Histogramm des hochgeladenen Passes mit Referenz-Histogrammen von allen Herausgebern vergleichen. Dem Pass wird dann der Herausgeber mit der höchsten Übereinstimmung zugewiesen. Diese Methode ist durch verschiedene Beleuchtung leicht fehleranfällig. Daher wurde durch ein Feature Matching versucht die Pässe dem Herausgeber zuzuordnen. Dabei werden alle Bildpunkte des Bildes mit Referenzpässen verglichen. Anschließend wird der Pass dem passenden Herausgeber zugeordnet.

Erkennung der Eckpunkte
Zur Erkennung der Eckpunkte wurde der Benutzer in einer ersten Version des Brillenpasslesers dazu aufgefordert, die Punkte manuell zu setzen. Später wurde ein Netz trainiert, welches automatisch die Eckpunkte erkennt. Dazu wird das gesamte Bild des Brillenpasses in Grid Cells eingeteilt. Für jede Grid Cell wird ein Score ermittelt, wie wahrscheinlich es ist, dass einer der Punkte in diesem Bereich liegt.

Hier geht's zum Brillenpassleser!
propose-ai.joelt.de

Über PROPOSE.AI

PROPOSE.AI ist eine Projektgruppe der Carl von Ossietzky Universität Oldenburg. Betreut wird das Projekt von der Abteilung Very Large Business Applications (VLBA) unter der Leitung von Prof. Dr.-Ing. Jorge Marx Gómez. PROPOSE.AI läuft seit April 2018 in Kooperation mit dem Online-Optiker Brille24.

12 Monate | 10 Studenten | 16h/Woche | Brille24/VLBA

Projektziel

Entwicklung einer AI Service Plattform für …
- Produktempfehlungen für Kunden,
- die Erleichterung des Einkaufsprozesses für Kunden,
- die Optimierung des Beschaffungsprozesses für Brille24 und
- die Optimierung des Marketingprozesses für Brille24

… auf Basis von Bild-, Shop- und Social Media-Daten.

Weitere Teilprojekte

Brillenberater
Das Ziel ist die Übertragung der persönlichen Beratung in den E-Commerce. Ein Kunde besucht den Brille24-Onlineshop, um sich eine neue Brille zu kaufen. Er ist von der breiten Masse an Angeboten überfordert und weiß nicht, welche Brille seinem Typ entspricht. Der Brillenberater ermittelt anhand von Gesichtsmerkmalen eine personalisierte, kleine Auswahl an Brillen, die individuell auf den Kunden zugeschnitten ist. Dies wird mit Hilfe von künstlicher Intelligenz und spezifischen Regeln umgesetzt, die die Intuition eines Optikers im stationären Handel simulieren.

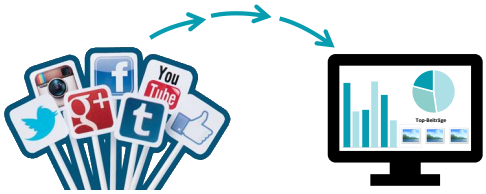
Trendanalyse
Der Fokus liegt bei diesem Projekt nicht auf dem Kunden, sondern auf den Mitarbeitern von Brille24. Diese sollen bei Entscheidungen in Einkaufs- und Marketingprozessen unterstützt werden. Dazu werden anhand von Bilddaten Trends erkannt und prognostiziert. Mittels künstlicher Intelligenz werden aus diesen Bildern Informationen zu Brillentyp, Rahmenfarbe und weiteren Eigenschaften extrahiert und mit zusätzlichen Daten verknüpft. Diese werden dann in einem Dashboard visualisiert.

Brillentinder
Mit dem Brillentinder soll der Kunde ebenfalls bei der Suche nach einer Brille unterstützt werden. Anders als beim Brillenberater liegt hierbei der Fokus auf dem persönlichen Geschmack des Kunden. Der Kunde kann ihm angezeigte Brillen bewerten und nähert sich so seinen präferierten Brillenmodellen.

PROPOSE.AI - Trendanalyse

Problemstellung & Ziel

In der heutigen Zeit der sozialen Medien teilen viele Menschen ihr Leben und ihren Style online. So werden auch oft Bilder von Brillen hochgeladen, da diese heute nicht nur einen medizinischen, sondern auch einen modischen Aspekt besitzen.
Um Prognosen oder einen Ist-Zustand für Brillentrends erkennen zu können, bedarf es einer umfangreichen Recherche und Analyse der sozialen Medien. Dabei ist es schwierig alle Erkenntnisse mit Zahlen zu untermauern. Aus diesem Grund wurde die Trendanalyse als Teilprojekt aufgenommen.
Ziel ist es, eine bessere Übersicht über die aktuelle Situation im Bezug auf Brillen zu erhalten. Das Dashboard soll diese und mögliche Entwicklungen abbilden. Auf diesem Weg können Trends sowie Negativtrends schneller und einfacher erkannt werden. Durch die gewonnen Erkenntnisse sollen der Einkauf und das Marketing unterstützt werden.



KI als Werkzeug

Um das oben genannte Ziel zu erreichen, kommt unter anderem künstliche Intelligenz zum Einsatz. Dabei liegt der Fokus zum einen darauf, der Maschine beizubringen Brillen auf Bildern zu identifizieren und zum anderen aus den erkannten Brillen wichtige, beschreibende Merkmale zu extrahieren.

Notwendige Schritte

1. Auswählen der entscheidenden Merkmale von Brillen
2. vorhandene Datensets sichten, aufbereiten und bereinigen
3. ggf. eigene Datensets aufbauen, d. h. Bilder mit Label versehen
4. 4a. künstliches neuronales Netz für die Object Detection entwerfen
   4b. künstliche neuronale Netze für die Erkennung der Merkmale entwerfen
5. Netze mit der zusammengestellten Datengrundlage trainieren
6. Ergebnisse auswerten
7. ggf. Anpassung der Netze

Der Weg zum Trenddashboard

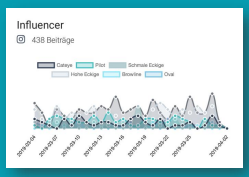
Der Fokus liegt bei der Trendanalyse nicht auf dem Kunden, sondern auf den Mitarbeitern von Brille24. Diese sollen bei Entscheidungen in Einkaufs- und Marketingprozessen unterstützt werden. Dazu werden anhand von Bilddaten Trends erkannt und prognostiziert. Mittels künstlicher Intelligenz werden aus diesen Bildern Informationen zu Brillentyp, Rahmenfarbe und weiteren Eigenschaften extrahiert und mit zusätzlichen Daten verknüpft. Diese werden dann in einem Dashboard visualisiert.

So funktioniert's

1. Bilder zu bestimmten Schlagwörtern werden gesammelt und in einer Datenbank gespeichert
2. Identifizierung aller Brillen auf den Bildern durch Object Detection
3. Cropping der erkannten Brillen

4. gecroppte Bilder durchlaufen die künstlichen neuronalen Netze zur Merkmalserkennung

5. ermittelte Merkmale zu den Brillenbildern werden in der Datenbank ergänzt
6. Bild-, Meta- und erkannte Merkmalsdaten werden in das Dashboard geladen und visualisiert



Dashboard-Ausschnitt

Was steckt dahinter?

Es wurde ein Datenpool mit insgesamt über **47.000** Bildern genutzt, der in **4** einzelne Datensets modifiziert wurde, um folgende Merkmale einer Brille mit künstlichen neuronalen Netzen identifizieren zu können:

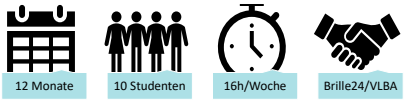
- Brillentyp,
- Brillenform,
- Rahmenart,
- Rahmenfarbe und
- Geschlecht des Brillenträgers.

Die entstandenen Netze erzielen durch das Training mit den Daten und weiterem Feintuning eine durchschnittliche Genauigkeit von **95%**.

Parallel dazu wurde ein Algorithmus entwickelt, mit dem passende Bilder zu ausgewählten Schlagwörtern gefunden und gespeichert werden. Zusätzlich wurde ein Dashboard-Konzept entwickelt und umgesetzt, welches die Ergebnisse der Netze in aufbereiteter Form darstellt.

Über PROPOSE.AI

PROPOSE.AI ist eine Projektgruppe der Carl von Ossietzky Universität Oldenburg. Betreut wird das Projekt von der Abteilung Wirtschaftsinformatik / Very Large Business Applications (VLBA) unter der Leitung von Prof. Dr.-Ing. Jorge Marx Gómez. PROPOSE.AI läuft seit April 2018 in Kooperation mit dem Online-Optiker Brille24.



Projektziel

Entwicklung einer AI Service Plattform für …

- Produktempfehlungen für Kunden,
- die Erleichterung des Einkaufsprozesses für Kunden,
- die Optimierung des Beschaffungsprozesses für Brille24 und
- die Optimierung des Marketingprozesses für Brille24

… auf Basis von Bild-, Shop- und Social Media-Daten.

Weitere Teilprojekte

Brillenpassleser

Der Brillenpassleser soll die Customer Experience des Kunden verbessern, indem ihm Arbeit abgenommen wird. Dabei kann der Kunde ein Foto seines Brillenpasses aufnehmen und hochladen. Die Werte werden dann durch den Brillenpassleser ausgelesen und automatisch in das Kundenkonto übertragen.

Brillenberater

Das Ziel ist die Übertragung der persönlichen Beratung in den E-Commerce. Ein Kunde besucht den Brille24-Onlineshop, um sich eine neue Brille zu kaufen. Er ist von der breiten Masse an Angeboten überfordert und weiß nicht, welche Brille seinem Typ entspricht. Der Brillenberater ermittelt anhand von Gesichtsmerkmalen eine personalisierte, kleine Auswahl an Brillen, die individuell auf den Kunden zugeschnitten ist. Dies wird mit Hilfe von künstlicher Intelligenz und spezifischen Regeln umgesetzt, die die Intuition eines Optikers im stationären Handel simulieren.

Brillentinder

Mit dem Brillentinder soll der Kunde ebenfalls bei der Suche nach einer Brille unterstützt werden. Anders als beim Brillenberater liegt hierbei der Fokus auf dem persönlichen Geschmack des Kunden. Der Kunde kann ihm angezeigte Brillen bewerten und nähert sich so seinen präferierten Brillenmodellen.

PROPOSE.AI - Architektur

CARL VON OSSIETZKY universität

VERY LARGE BUSINESS APPLICATIONS

BRILLE24 Research

Skalierbarkeit		Erweiterbarkeit
Horizontale Skalierbarkeit aller Dienste	Lastverteilung als integraler Bestandteil	Basisbibliothek kapselt die fachliche Logik
I. d. R. alle Dienste zustandslos	Kein extra Load-Balancer benötigt	Entkopplung von Kommunikationslogik und fachlicher Logik

Architekturprinzipien

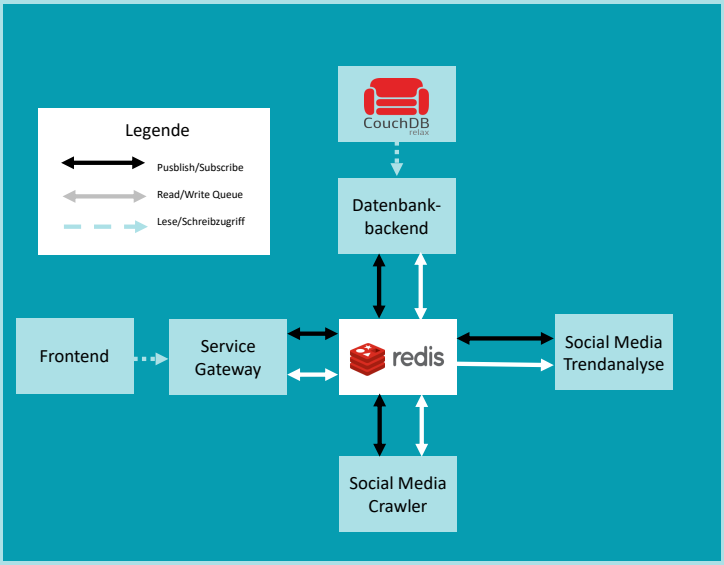
Reliable Queue Pattern für Nachrichten

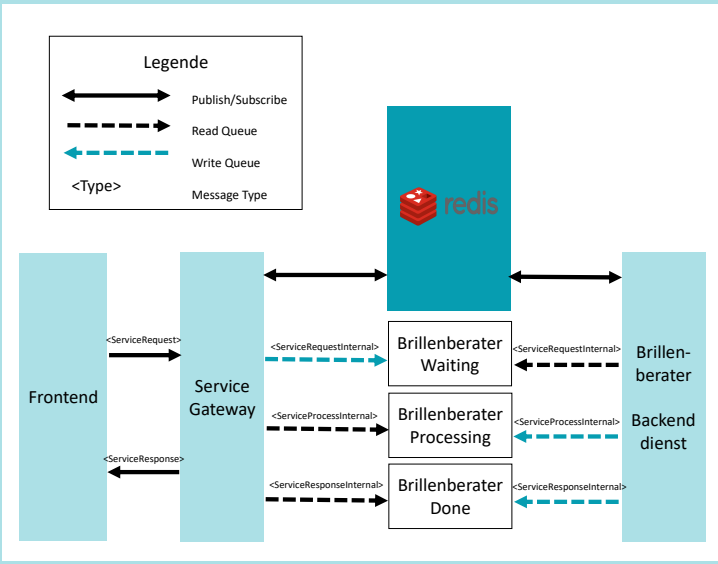
- Nachrichten werden erst aus der Quellenque gelöscht, wenn sie vorher in eine andere Queue kopiert wurden
  - Redis bietet dafür z.B. RPOPLPUSH als atomare Operation
- Verhindern Datenverlust im Fehlerfall
- Asynchrone Nutzung möglich

Leichtgewichtigkeit der Plattform

- Publish/Subscribe Events vermeiden aktives Polling
- „Light Polling“ nur im Service Gateway

Servicekommunikation





redis

Redis ist eine In-Memory Datenbank mit einfacher key-value Datenstruktur. Es ist der Familie der NoSQL-Datenbank zuzuordnen. Gegenüber relationalen Datenbanken besteht der Vorteil in der schnelleren Verarbeitung.

CouchDB relax

CouchDB ist ein Datenbankmanagementsystem von Apache. Es verknüpft das einfache Datenmodell einer dokumentenorientieren Datenbank mit der Skalierbarkeit und Leistung einer professionellen relationalen Datenbank.

Services

Frontend

- Trendanalyse Dashboard
- Frontend
- Labeltool Frontend
- Labeltool Frontend

Zentrale Dienste

- Service Gateway
- Datenbankdienst

Fachliche Dienste

- Brillentinder
- Brillenpassleser
- Brillenberater
- Social Media Dashboard-Service
- Social Media ML-Service
- Social Media Crawler

Werkzeuge

- Evaluation-Service
- Labeltool

Abschließende Erklärung

Wir versichern hiermit, dass die vorliegende Arbeit selbständig und ohne fremde Hilfe angefertigt wurde, und dass alle von anderen Autoren wörtlich übernommenen Stellen wie auch die sich an die Gedankengänge anderer Autoren eng anlegenden Ausführungen dieser Arbeit besonders gekennzeichnet und die Quellen zitiert sind.

gezeichnet
Albrecht, Jedebrock, Lohmann, Meyer, Oeltjen, Otten, Riedel, Tiemerding, Witte, Zurborg

Oldenburg, den 10. Mai 2019