Carl von Ossietzky Universität Oldenburg
Faculty II - Department of Informatics
Digitalisierte Energiesysteme

Effective constraint satisfaction for Deep Reinforcement Learning approaches to Optimal Power Flow problems

Bachelor thesis

submitted by

Fenno Boomgaarden

born on 25.09.1997 in Aurich

First Examiner:	Prof. Dr.-Ing. Astrid Nieße
Second Examiner:	M.Sc. Thomas Wolgast
Supervisor:	M.Sc. Thomas Wolgast

Oldenburg, May 25, 2022

Abstract

Optimal power flow is a class of complex problems of modern energy systems, aiming to control all energy flows of the system in an optimal way by minimizing an objective function without violating the systems constraints. Real world applications of optimal power flow are often non-linear and very time consuming to solve using conventional approaches. Deep reinforcement learning approaches have shown promising results to solve optimal power flow problems in real time, but commonly use the constraint satisfaction strategy of modifying the reward with fixed penalty functions. This thesis demonstrates by experiment how fixed penalties suffer from a trade-off between constraint satisfaction and objective optimality on more complex energy systems. To provide a more effective constraint satisfaction strategy, the Constrained Twin Delayed Deep Deterministic Policy Gradient algorithm is proposed as a sample efficient off-policy safe deep reinforcement learning algorithm, using the method of a learned lagrange multiplier. The proposed algorithm slowly adjusts a lagrange penalty multiplier to avoid the policy to become too conservative and shows how guarantees for constraint satisfaction within a given error threshold are possible. As shown by experiment, this method avoids the trade-off problem and achieves both high objective performance and constraint satisfaction.

Acknowledgements

I would like to express my gratitude to all, who supported me during my studies and who helped review this bachelor thesis.
I also like to thank my supervisor Thomas Wolgast for his very helpful supervision, feedback and discussions.

.

A very special thanks goes out to my former IT-teachers Arend Ihnen and Axel Schudak for teaching me programming and introducing me to neural networks at very young ages.

Contents

Abbreviations

A2C	Advantage Actor Critic
AC	Alternating Current
ANN	Artificial Neural Network
CMDP	Constrained Markov Decision Process
CPO	Constrained Policy Optimization
CSAC	Constrained Soft Actor Critic
CSV	Comma Seperated Values
CTD3	Constrained Twin Delayed Deep Deterministic Policy Gradient
DC	Direct Current
DDPG	Deep Deterministic Policy Gradient
DL	Deep Learning
DNN	Deep Neural Network
DRL	Deep Reinforcement Learning
MDP	Markov Decision Process
OODN	Optimal Operation of Distribution Networks
OPF	Optimal Power Flow
PPO	Proximal Policy Optimization
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
RNN	Recurrent Neural Network

SAC	Soft Actor Critic
SDRL	Safe Deep Reinforcement Learning
SRL	Safe Reinforcement Learning
TD3	Twin Delayed Deep Deterministic Policy Gradient
TRPO	Trust Region Policy Optimization

1 Introduction

1.1 Motivation

Electrical power systems are facing large difficulties with the increasing demand for electricity and the transition to renewable energies and decentralized energy markets [1]. The Optimal Power Flow (OPF) problem formalizes this difficulties mathematically, aiming to control all power flows of an energy system in an optimal way by minimizing an objective function, subject to the systems constraints and operating limits. OPF problems are often non-linear and non-convex and therefore very time consuming to solve with conventional methods, especially because the constraints increase the complexity significantly. With the progress of machine learning in the recent years, Deep Learning (DL) has shown promising results for solving OPF in real time [2]. Especially Deep Reinforcement Learning (DRL) algorithms show great potential for this problem, as they can provide nearly optimal solutions to OPF in orders of magnitude less time in a model free manner, training an agent by trial and error to maximize a given reward. Because constraint violations can lead to unpredictable costs and damage to the systems involved, a sufficient constraint satisfaction strategy is necessary. However, most of the previous DRL approaches to OPF augment the reward with a fixed penalty function, making constraint violations unattractive to the agent. Two methods of fixed penalty functions are commonly used: the summation method penalizes constraint violations with a fixed linear penalty factor and the replacement method uses different reward functions for states with violations and states without. According to the literature, the summation method is expected to result in either suboptimal policies or unacceptable amounts of constraint violations [3], [4]. Therefore, this thesis investigates the relevance of this problem for the summation method, tests if the replacement method is a suitable alternative and compares the two methods with promising Safe Reinforcement Learning (SRL) strategies from the literature.

1.2 Objective

The main objective of this thesis is to answer the question, how effective constraint satisfaction can be achieved or even guaranteed for DRL approaches to OPF problems. Effective

constraint satisfaction in this context means to not only ensure constraint satisfaction, but to also achieve the highest objective performance that is possible within the constraints. As mentioned before, previous DRL approaches to OPF use fixed penalty functions to prevent constraint violations. The summation method modifies the reward with a fixed linear violation penalty factor, which can not guarantee constraint satisfaction and also raises the question, how an optimal penalty factor can be selected and if the learned policy can reach optimal objective performance within the constraints. As mentioned by Wang et al. [3], a penalty factor set too high can lead to a decreased objective performance, because the policy is trained to be too conservative. A smaller penalty on the other hand can result in more frequent constraint violations, which is also not acceptable. Having this trade-off between objective performance and sufficient constraint satisfaction, balancing the penalty factor constitutes an optimization problem that has to be solved experimentally for every new problem the approach is applied to. Although the trade-off is highly expected as a problem of the summation method, it is not clear how relevant it actually is within the OPF application. The first research question of this work is therefore:

Research Question 1:
How relevant is a trade-off between objective performance and constraint satisfaction for the fixed penalty DRL approaches to the OPF problem?

A second common strategy used in the literature is the replacement method, using different rewards for states with violations and states without. Because the rewards of states with violations are in orders of magnitude smaller than the rewards of states without violations, constraint satisfaction is always more important to the agent than optimal objective performance. Therefore it is plausible that the replacement method also leads to policies that are too conservative, comparable to the summation method with a very high penalty factor. The second research question therefore is:

Research Question 2:
Which advantages and disadvantages regarding effective constraint satisfaction does the replacement method have within the OPF problem, compared to the summation method?

The possible problem of the trade-off leads to the question, if there are more effective strategies that can improve or even provide guarantees for the constraint satisfaction, which can be used in DRL algorithms. As shown in the literature, Safe DRL (SDRL) approaches have successfully solved comparable problems in the past, especially with strategies based on the lagrange relaxation method and applications of the CPO algorithm [3], [4]. Therefore the third research question of this thesis is:

Research Question 3:
Which advantages and disadvantages regarding effective constraint satisfaction do SDRL approaches have within the OPF problem, compared to the fixed penalty methods?

1.3 Structure

In Chapter 2, the background for OPF, DRL and SRL is provided with focus on the algorithms and theoretical foundations most relevant to this thesis. Afterwards, in Chapter 3, previous DRL approaches to OPF are surveyed and their different constraint satisfaction strategies are analysed, followed by a survey of the most promising SDRL approaches to comparable problems within the energy systems context. Chapter 4 explains the approach and the algorithms and methods used in this thesis in detail. Afterwards, the three conducted experiments are described in Chapter 5, followed by the collected results of the experiments in Chapter 6. The results are then discussed in Chapter 7, reflecting on the approach and answering the research questions. The thesis is then concluded in Chapter 8 with a short summary of the research done and its findings.

2 Background

This chapter provides the theoretical background and foundations for the topics and approaches covered in this thesis. In Section 2.1, the OPF problem is introduced with focus on its computational difficulties in regard to the optimization objective and constraint satisfaction. The next two sections Section 2.2 and Section 2.3 cover the core principles and algorithms of DL and Reinforcement Learning (RL), building the foundation for the most commonly used and state of the art DRL algorithms covered in Section 2.4. Finally, in Section 2.5 an overview of SRL concepts and strategies is provided. The overview part of the first four sections as well as the SRL section have been previously elaborated within the proposal of this thesis and are reused in a sligthly modified form.

2.1 Optimal Power Flow

The following section summarizes the OPF problem with main emphasis on the constraints as described by Frank et al. [1]. OPF is an optimization problem for electric power control, that can represent the increasing complexity of modern electrical power systems, caused by the uncertainty introduced with the deregulation of electricity markets and the introduction of renewable energies. In most applications, the objective of OPF is to optimize the control variables of the systems power flow for minimal generation costs or minimal network losses without neglecting the systems constraints or control limits, ensuring safe and reliable operation of the power system. The standard formulation of the OPF problem consists of the objective function f to be minimized and a set of equality constraints g and inequality constraints h for control variables u and a current systems state x.

$$\begin{aligned} \min \quad & f(u, x) \\ s.t. \quad & g(u, x) = 0 \\ & h(u, x) \leq 0 \end{aligned}$$

The objective function, constraints, control variables and system state representation vary between different OPF formulations and applications. Because in many cases the objective

function as well as the constraints are nonlinear and non-convex, the problem becomes increasingly challenging to solve with better representations of the systems involved.

2.1.1 Variables

The control variables are usually a subset of the state variables combined with device control settings, for example real and reactive power of the generators or transformer tap ratios. State variables are a continuous representation of the electric state of the system, including bus voltage magnitude, bus voltage angle, and real and reactive power injections at the busses.

2.1.2 Objective function

Usually, the objective function contains some notion of generation costs but it can also include system losses, power quality, planning costs or even the environmental impact of the power generation. With the generation costs being the most common objective function, OPF can extend the classic economic dispatch problem to not only control the generation units to dispatch but all power flows of the system. The generation cost can then be represented by quadratic or piecewise linear cost curves. Power quality can also be ensured as part of the objective function by minimizing voltage deviation.

2.1.3 Equality constraints

The equality constraints consist of the power flow network equations, in their polar or rectangular form, and other balancing constraints. The power flow equations in their complete form represent AC power flow of the system and are nonlinear as well as non-convex. A common approach to the problem is a simplification to a linear DC power flow that is much easier to solve but far less accurate. Therefore, in many applications, a full AC power flow solution is necessary.

The more often used polar form of the AC power flow equations depends on state variables for voltage magnitude $|V|$ and voltage phase angle δ. This form has the advantage of a strong coupling of real power P with voltage angle δ and reactive power Q with voltage magnitude $|V|$. Y_{ik} is the ik-th element of the bus admittance matrix and θ_{ik} is the angle of the ik-th element of the bus admittance matrix of the power system.

$$P_i = \sum_{k=1}^{N} |V_i||V_k||Y_{ik}| \cos(\delta_i - \delta_k - \theta_{ik}) \tag{2.1}$$

$$Q_i = \sum_{k=1}^{N} |V_i||V_k||Y_{ik}| \sin(\delta_i - \delta_k - \theta_{ik}) \tag{2.2}$$

The rectangular form has the advantage of constant second partial derivatives and comes without the trigonometric function sin and cos, depending on the real and imaginary voltage components R and F as state variables. G_{ik} and B_{ik} are the real and imaginary parts of the bus admittance matrix Y.

$$P_i = \sum_{k=1}^{N} G_{ik}(E_iE_k + F_iF_k) + B_{ik}(F_iE_k - E_iF_k) \tag{2.3}$$

$$Q_i = \sum_{k=1}^{N} G_{ik}(F_iE_k - E_iF_k) + B_{ik}(E_iE_k + F_iF_k) \tag{2.4}$$

2.1.4 Inequality constraints

Inequality constraints represent upper and lower control limits as well as security and stability constraints, preventing physical damage to the systems involved. The control limits can include the generation limits of the power generators, the bus voltage and branch flow limits and operational limits of other electrical systems. They can also include limits on the environmental impact of the power system.

2.2 Deep Learning

DL is a part of machine learning that utilises Deep Neural Networks (DNN) to learn patterns in high dimensional, unstructured raw input data. Mainly used in supervised but also unsupervised learning methods, a DNN can learn, recognize and generalize abstract patterns in large datasets, making it highly applicable for speech recognition, computer vision and natural language processing. DNNs consist of multiple layers of virtual neurons, processing units, that imitate the mathematical properties of biological neurons. A DNN consists of an input layer, multiple in-between layers, called hidden layers, and an output layer. Every hidden layer processes and abstracts the information from the previous layer and puts it forward to the next layer. DNNs can be trained on datasets using a gradient descend algorithm, which can use backpropagation to propagate the network losses backwards

through the network and adjust the parameters causing the errors, to increase the networks accuracy. [5]

2.2.1 Artificial Neural Network

The following subsection is based on the information on Artificial Neural Networks (ANN) provided by the book "Artificial Neural Networks - A Practical Course" [6].

Artificial neurons, the core concept behind ANNs, have been first described by Warren McCulloch and Walter Pitts in 1943 [7], proposing a simplified computational model for biological neurons with the McCulloch-Pitts Neuron. Later, Frank Rosenblatt introduced the Perceptron [8] and a learning algorithm based on Hebb's rule [9], building the basis for modern ANNs.

An artificial neuron is a simple computational unit, that represents the most fundamental information processing of biological neurons in a highly simplified form. It consists of a vector of *weights* $w = [w_1, ..., w_m]$ representing the synaptic weights and a *threshold* or *bias* value θ for the threshold of the axon hillock, takes a vector of *input* values $x = [x_1, ..., x_m]$ representing the dendrites, and has one single *output* value y for the axon of a biological neuron. The input values are multiplied by the weights and are then added up to an *activation* u.

$$u = w^\top \cdot x - \theta = \sum_{i=1}^{n} x_i w_i - \theta \tag{2.5}$$

The output y of the neuron is then determined by an *activation function* $g(u)$, for example the Heaviside step function for the most simple perceptron.

$$g_{heav}(u) = \begin{cases} 1 & \text{if } u > 0, \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

From a mathematical perspective, these binary perceptrons are binary linear classifiers, so they can only represent linearly separable classes if used in a single layer. Being fundamental building blocks of machine learning, perceptrons can *learn* linear classifications by training on a set of training examples. For binary output perceptrons, a training example is a pair of an input value vector $x^k = [x_1^k, ..., x_m^k]$ and an expected binary output a^k. The training algorithm for binary perceptrons loops over all examples for a given number of iterations,

calculates the error $a^k - y^k$ for each example and then adjusts the weights of the perceptron according to their contribution to the error. α is the learning rate, a small value $\alpha \in (0, 1)$.

$$\Delta w_i = \alpha(a^k - y^k)x_i^k \tag{2.7}$$

If the examples are linearly separable sets, this algorithm is proved to converge to the linear classification given infinite update steps, otherwise it converges to a linear classification with a minimal error. [9]

Binary classification has its limitations, when applied to real world examples. To abstract the activation frequency and to deal with uncertainty, continuous and more probabilistic activation functions can be used, such as the sigmoid and the tanh activation functions:

$$g_{sig}(u) = \frac{1}{1 + e^{-u}} \tag{2.8}$$

$$g_{tanh}(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}} \tag{2.9}$$

The sigmoid function has the advantage of being easily differentiable, allowing neurons with this activation function to be trained with a gradient descend algorithm using the *Delta Rule*:

$$\Delta w_i = \alpha(a^k - y^k)g'(u^k)x_i^k \tag{2.10}$$

Nowadays it has been shown that the Rectified linear unit (ReLU) outperforms the sigmoid activation function in many DL applications, being faster to calculate, even more simple to differentiate and being less prone to the vanishing gradient problem other activation functions have [10].

$$g_{ReLU}(u) = \max(u, 0) \tag{2.11}$$

2.2.2 Deep Neural Networks

Because real world problems are often very complex and non linear, the single layered perceptron is not very useful for most applications. Therefore DNNs, large, multi layered neural networks, are used, that can learn non linear behaviour. In a DNN, the outputs of

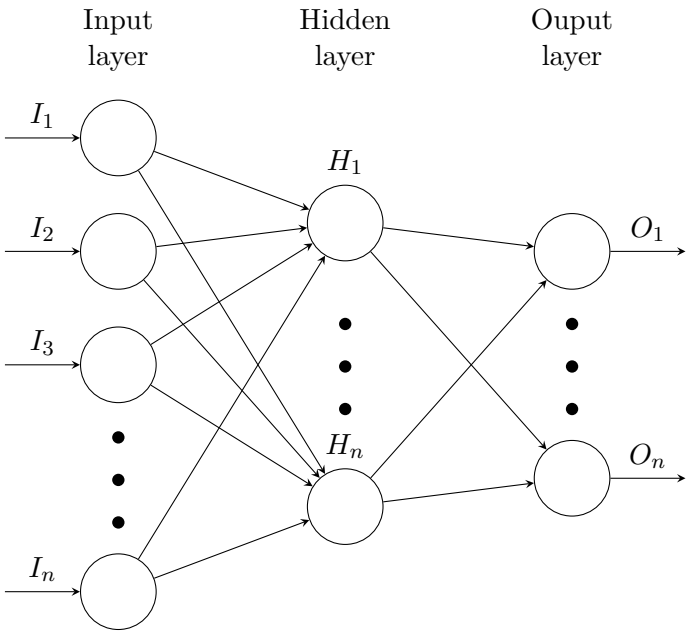


Figure 2.1: Feed forward DNN with one hidden layer

a neuron can be the inputs of another neuron, building multiple processing layers on top of each other.

In Feed-forward DNNs the neurons are sorted in layers such that every neuron is only connected with its inputs to the outputs from the previous layers, as shown in Figure 2.1. Connections that skip layers are sometimes allowed and are called shortcuts. Feed-forward DNNs can be trained in a gradient descend algorithmn, using the backpropagation method [11], that generalizes the *Delta Rule* to DNNs by automatic differentiation. The chain rule is applied, to propagate the error backwards through the network and adjust the weights accordingly to their influence on the error. This allows every layer of the network to learn a layer of abstraction, by learning patterns in the classifications output from the previous layer, resulting in non linear and highly complex behaviour. In contrast to feed-forward DNNs, other DNNs do allow recursion to happen and are called Recurrent Neural Networks (RNN). In a RNN a neuron can also feed information to itself or to previous neurons, such that loops can emerge. RNNs are more difficult to train, because their structure is less straightforward, making training more difficult. Nevertheless, RNNs have shown outstanding results in complex time series classification problems, such as speech recognition. [5]

2.3 Reinforcement Learning

This section is based on the book "Reinforcement learning: An introduction" by Sutton and Barto [12]. RL describes, how an agent can learn from interactions with its environment to reach a given goal. The problem is described as a finite Markov Decision Process (MDP), represented by the tuple (S, A, P, R). As shown in Figure 2.3, the environment at time t is represented by a state $s_t \in S$, that can be fully or partially observed by the agent. The agent then chooses an action $a_t \in A$ to perform next in the environment, according to its policy function π that represents the agents behaviour as a deterministic $a_t = \pi(s_t)$ or stochastic $\pi(a|s) = \mathbb{P}[a_t = a \mid s_t = s]$ policy for every state. In the following, the policy is assumed to be stochastic for better generalization. The taken action a_t influences the probability of the next state $s_{t+1} \in S$, represented with the transition function $P_a(s, s') = \mathbb{P}[s_{t+1} = s' \mid s_t = s, a_t = a]$. The agent then receives a reward $r_t = R_{a_t}(s_t)$ for its action, representing its progress towards the goal. There are three general types of RL algorithms, Value-Based, Model-Based and Policy-Based, but combinations are also possible.

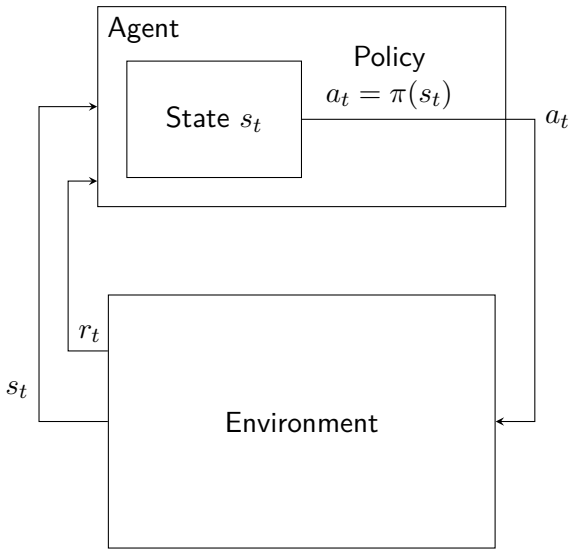


Figure 2.2: RL agent-environment relationship. The agent observes the state s_t from the environment, gets a reward r_t for the previous action and uses its policy π to select a new action a_t to execute next.

2.3.1 Value Functions

RL algorithms adjust the agents behaviour by trial and error to maximize the return, that is the sum of the rewards the agent accumulates over time. Instant rewards and far future

rewards can be balanced by a discounting factor $\gamma \in [0, 1)$ resulting in a discounted return G_t, such that a low discounting factor represents more focus on immediate rewards and a high discounting factor represents more awareness of far future rewards.

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2.12}$$

The expected discounted return of a state s is represented recursively with the value function $v_\pi(s)$, depending on the policy of the agent. This is called the Bellman Expectation Equation.

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi \left[G_t \mid s_t = s \right] \\ &= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right] \\ &= \mathbb{E}_\pi \left[r_{t+1} + \gamma \, v_\pi(s_{t+1}) \mid s_t = s \right] \end{aligned} \tag{2.13}$$

This concept can also be applied to state-action values, representing the expected discounted return if action a is taken in state s. This is also referred to as the Q-function and its values are referred to as Q-values.

$$\begin{aligned} q_\pi(s, a) &= \mathbb{E}_\pi \left[G_t \mid s_t = s, a_t = a \right] \\ &= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right] \\ &= \mathbb{E}_\pi \left[r_{t+1} + \gamma \, v_\pi(s_{t+1}, a_{t+1}) \mid s_t = s, a_t = a \right] \end{aligned} \tag{2.14}$$

The optimal value function v_* and the optimal Q-function q_* represent the best possible values and Q-values for states and actions.

$$v_*(s) = \max_\pi v_\pi(s) \tag{2.15}$$

$$q_*(s, a) = \max_\pi q_\pi(s, a) \tag{2.16}$$

An RL algorithm uses this equations to make adjustments to the agents policy for it to converge with infinite time to the optimal policy π_*, which would be always choosing the actions with the highest possible Q-values.

$$\pi^*(a|s) = \begin{cases} 1 & \text{if } a = \arg\max_{a' \in A} q_*(s, a'), \\ 0 & \text{otherwise} \end{cases} \tag{2.17}$$

There are model-based and model free RL algorithms. Model based algorithms learn a model of the environment that can predict the consequences of the possible actions the agent can take in each state. With a good model of the environment, the agent can use search algorithms to select action with the most promising future state trajectories.

There are also model-free algorithms, that can learn optimal behaviour without the need of a model. Model-free algorithms can be policy-based or value-based or a combination of both. Policy-based algorithms strive to learn the optimal policy itself while value based algorithms learn estimate functions for the state-value or action-value. In value-based strategies the value estimate can then be used in a $\epsilon - greedy$ policy, choosing a random action with a probability of ϵ and choosing the action with the maximal action-value otherwise.

A common strategy of RL is temporal difference learning, a model-free and value-based method that updates its value function after every interaction with the environment. The core priciple of temporal difference learning is *bootstrapping*: The reward received after the taken action is combined with the value estimation of the current state to get a temporal difference error gradient to update the estimation of the last state. The Actor-Critic method applies temporal difference learning to a combination of value-based and policy-based RL, learning both a policy (the Actor) and the value estimate (the Critic). After each interaction with the environment the Critic provides a value estimate that is used to calculate the temporal difference error, which can then be used to update both Actor and Critic.

2.4 Deep Reinforcement Learning

DRL is a combination of RL and DL, representing parts of the agent by DNNs that can detect patterns in the observations of the agent and learn complex function approximations for the optimal policy, value function or model using gradient ascend. Figure 2.3 shows how the policy of a DRL agent is represented by a DNN, choosing actions from a given state observation.

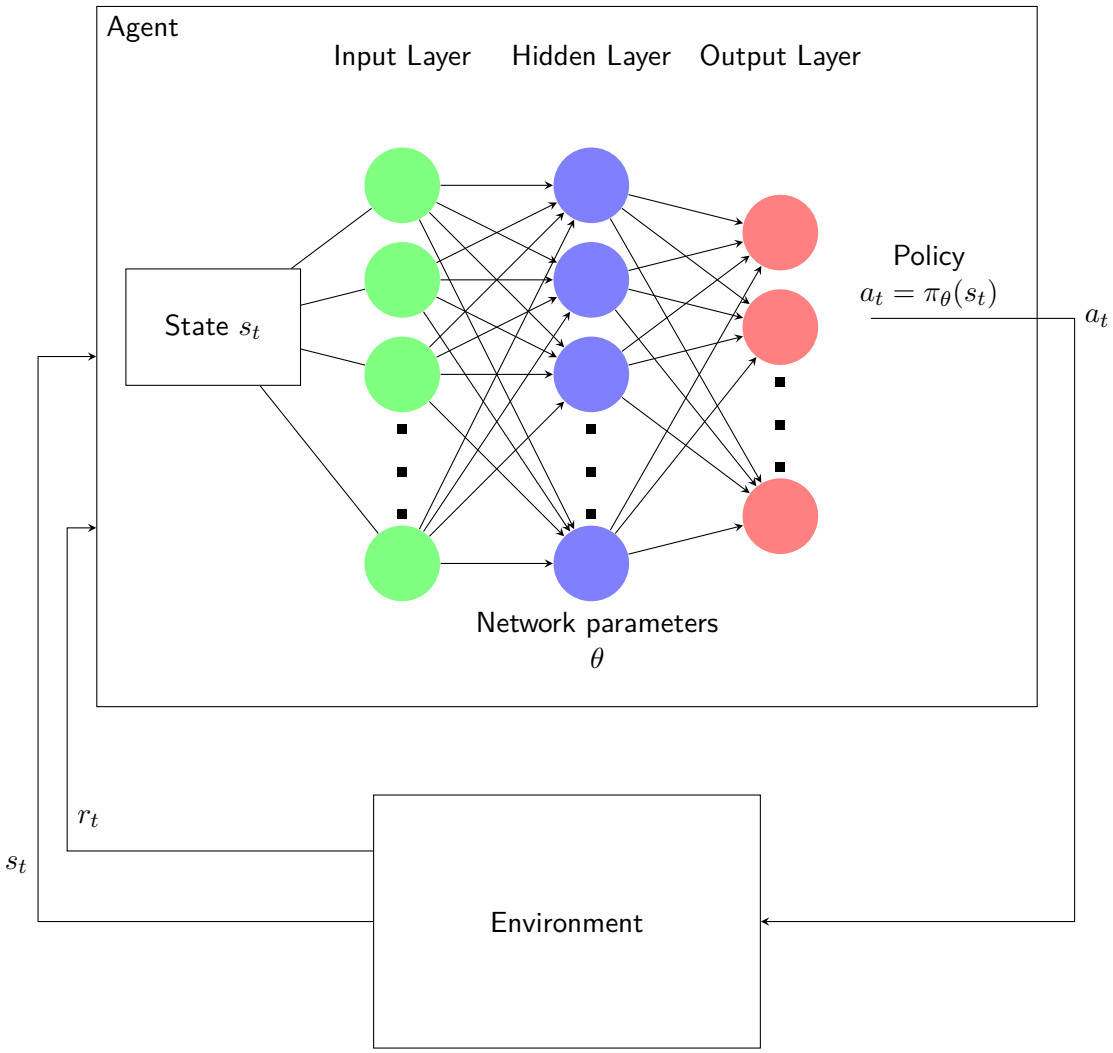


Figure 2.3: Basic DRL setup. The agents policy is represented by a actor neural network, that takes the current state as input and produces the agents next action as output. The network is trained using the reward with a DRL algorithm, often using additional DNNs to derive the policy gradient.

Many state of the art DRL algorithms are based on the Actor-Critic method, representing both Actor and Critic by a DNN. In the following section, some state of the art DRL algorithms are explained that are relevant to this thesis.

2.4.1 Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient (DDPG) is an Actor-Critic DRL algorithm with Critic and Actor represented by a DNN. It is based on Deep Q-Learning, with the Critic DNN learning the Q-function using the reward by application of the Bellman Equations. The Actor then improves the policy with a gradient ascend method using the gradient of the critics action-value output. DDPG is useful in environments with continuous action spaces, where $\epsilon - greedy$ is not applicable because the action with the highest approximated action-value is difficult to search for. [13]

2.4.2 Twin Delayed DDPG

Because DDPG is prone to the problems of value overestimation, suboptimal policies and divergent behaviour, Twin Delayed DDPG (TD3) has been introduced to solve this issues. In TD3, three mayor adjustments are made to the original DDPG algorithm. The first is called Clipped Double-Q Learning, learning two Q-functions instead of one and using the smaller Q-value for the Bellman error loss, adressing the problem of overestimation. The second adjustment is, to update the policy less frequently than the Q-function. Finally with target policy smoothing, noise is added to the target action to reduce the expoitation of Q-function errors by the policy. [14]

2.4.3 Soft Actor Critic

Another algorithm invented to solve the problems of DDPG is Soft Actor Critic (SAC), also using the trick of Clipped Double-Q Learning. However, instead of using target policy smoothing SAC trains a stochastic policy that maximizes both the policy entropy and expected return, with the balance determined by a temperature parameter. It has been shown that maximal entropy policies substantially improve robustness and exploration, because the policy learns more diverse behaviours. [15]

2.4.4 Proximal Policy Optimization

Proximal Policy Optimization (PPO) is a policy gradient method based on Trust Region Policy Optimization (TRPO). In policy gradient methods a DNN is trained in a policy-based way towards an optimal policy, using stochastic gradient ascend on the policy gradient. The

most common policy gradient is calculated as follows, with θ being the policy parameters of the DNN and $\hat{A}_t$ being the advantage of a_t, that is the difference between action-value and value. [16]

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta(a_t \mid s_t) \hat{A}_t \right] \tag{2.18}$$

Because multi-step updates of this vanilla policy gradient method can often lead to destructively large policy updates, learning from batch experience is often not possible. TRPO solves this problem by limiting the policy update step to a safe stepsize using a constraint at the cost of increasing the algorithms complexity and incompatibilities with noise and parameter sharing [17]. PPO therefore simplifies this approach by introducing a clipped surrogate objective. [16]

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{2.19}$$

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right] \tag{2.20}$$

2.5 Safe Reinforcement Learning

This section covers the concepts of SRL as described by García and Fern [18]. Although RL and especially DRL algorithms can learn nearly optimal policies for MDPs by interacting with the environment, it usually can not guarantee the safety of the agent and its surroundings in high risk environments. For example, a physical robot is expensive to build and should not be allowed to cause harm or damage to itself or its environment. For this situations, SRL strategies have been developed, to provide better damage prevention in this safety-critical applications.

There a two major strategies used in different forms for SRL. The first strategy modifies the agents optimization criterion to contain some notion of risk. The second strategy modifies the exploration process of the agent with external knowledge or a risk metric.

For the first strategy, there are three major modifications in use: The Worst-Case Criterion, the Risk-Sensitive Criterion, the Constraint Criterion. Instead of optimizing for the expected discounted cumulative reward, the Worst-Case Criterion optimizes for the expected discounted cumulative worst possible reward, always assuming the worst state transition to happen. The Risk-Sensitive Criterion allows the desired level of risk to be controlled by a scalar parameter in the optimization criterion, using an exponential utility function or a weighted sum of expected cumulative reward and risk. The Constraint Cri-

terion might be especially relevant to this thesis because it formulates the problem as a Constrained Markov Decision Process (CMDP). In a CMDP the objective is to optimize for a given goal without violating the given constraints, restricting the policy space to a subset of allowed policies. A possible way to solutions for the CMDP is the lagrangian relaxation of the constrained optimization criterion into an unconstrained lagrangian expression, including the constraints in the optimization criterion while preserving the original objective.

For the second strategy, the exploration process can be modified with external knowledge. Initial knowledge can be used to bootstrap the learning agent, to switch to a fully greedy exploration, exposing the agent to the most relevant state and action spaces. Exploration can also be modified by deriving a policy from a finite set of demonstrations. Instead of random examples, the agent learns from a set of given examples. It is also possible to use external knowledge in form of a so called teacher, that follows the same goal as the agent and assists the agent with training. Furthermore a risk metric can be used for risk-directed exploration, determining the exploration probability for different actions by the risk of that actions.

3 Related Work

3.1 DRL approaches to OPF

Solving OPF problems with DRL is an open field of research. Cao et al. [19] use PPO to provide OPF control decisions in real time for a IEEE 33-bus system. In this approach, the power system is modified to contain renewable energy generators and storage devices, to better match the conditions of a real world application. It outperforms Stochastic Programming and Double Deep Q-Learning but points out that ensuring constraint satisfaction with SDRL needs to be studied in later works. To prevent constraint violations, Yan et al. [20] successfully apply the Lagrangian relaxation method to the reward, transforming the constrained OPF formulation into an unconstrained approximation objective using Lagrange multipliers, resulting in penalties that do not conflict with the optimization goal. This approach uses the DDPG algorithm with the interior point method for the gradient approximation during training on a IEEE 118-bus system. Another approach is presented by Zhou et al. [21], using PPO to solve AC-OPF without violating constraints. In this approach, the agent is trained with the PPO algorithm, setting a penalty linear to the amount and magnitude of constraint violations, replacing the optimization reward if violations occurred. When using Imitation Learning for the initialisation of the DNN, this has shown to prevent constraint violations while improving computation time, compared to the interior-point method. Woo et al. [22] apply the TD3 algorithm to solve the AC-OPF problem in real time, demonstrated on a IEEE 118-bus system and using Gaussian noise to represent uncertainties of renewable energy sources. The presented approach uses linear penalties added to the reward for constraint violations, which can be detected at train-time by solving the power flow equations. The results are compared to a MATPOWER AC-OPF solver on a IEEE 118-bus system and match its performance at 30 times less computation time. Nie et al. [23] use the DDPG algorithm to solve the OPF problem in real-time on a small IEEE 9-bus system. In this approach, a quadratic penalty is added to the reward to prevent constraint violations. The approach provides nearly optimal OPF solutions compared to the MATPOWER OPF solver. Zhou et al. [24] solve the OPF problem using PPO

on the Illinois 200-bus system with wind generation variation, replacing the optimization objective with linear penalties if constraint violations occur.

All approaches examined use penalties as a strategy to avoid constraint violations. Two major strategies, how a penalty can be applied to the reward function, can be identified. The first strategy will be referred to as the *summation* method, which sets different penalty coefficients for constraint violations added to the cost function, which are implemented as lagrangian multipliers in [20], linear coefficients in [22] and quadratic coefficients in [23].

The second strategy, used in [21] and [24], will be referred to as the *replacement* method and has different reward functions for different operating states. If the operational state is valid, the reward is set as the optimality reward using the generation costs subtracted from a high value (1000 for [24]). If constraints are violated, the linear constraint violation penalties are replacing the optimality reward. If the solution is diverged, a super high penalty of e.g. -5000 is used.

The Lagrangian based method in [20] utilises the Constrained Optimization Criterion mentioned in Section 2.5, using a Lagrangian expression to transform the constrained optimization criterion into an unconstrained expression. Therefore, [20] can actually be classified as a SDRL approach with the penalties more sophisticated, but this is not explicitly mentioned by the paper.

Although OPF problems and SRL are both open and wide fields of research, there is no other SRL approach to OPF yet. However, as described in the next section, SDRL has been successfully applied to other problems related to power systems.

3.2 SDRL approaches to CMDPs similar to OPF

In this section, different promising approaches of SRL to problems with similarities to the OPF problem are reviewed, providing a basis to select from when choosing the approach and algorithms used in the SRL implementation of this thesis. First of all, the criteria and similarities to look for have to be defined. The main safety challenge of the OPF problem are its constraints, making it describable as a CMDP. Therefore, the main focus of this analysis is, to find SRL approaches that specifically aim to solve CMDPs with DRL, preferably within the energy systems context. Currently, there are three papers that apply SDRL to CMDP problems of energy systems, using variations of two very promising approaches. The first approach is the Constrained Criterion mentioned in [18], using lagrangian relaxation to transform the constrained objective into an unconstrained one. The second approach is based on the Constraint Policy Optimization (CPO) algorithm, a PPO variation dedicated to solve CMDPs with PPO.

As found out in the last section Yan et al. [20] provided the only SDRL approach to the OPF problem so far, using the Lagrangian relaxation method with DDPG to transform the constraints into penalty coefficients in the objective function.

In [3], a SDRL approach is successfully used to solve the Volt-VAR problem, which can be a partial problem of OPF as it aims to control voltage levels and reactive power flow in power distribution systems subject to system constraints. The paper proposes Constrained Soft Actor Critic (CSAC), a novel policy gradient method, that applies SAC to CMDPs. CSAC combines the advantages of the SAC algorithm with the method of Lagrange multipliers, relaxing the constrained CMDP objective into an unconstrained one. To be able to calculate the Lagrange multipliers, the algorithm involves additional action-value and state-value functions, separate from the optimality value functions, that represent the constraint violations as their own state-values and action-values. Being a SAC method, the algorithm enforces maximum policy entropy, the Q-functions for both optimality and constraints are represented twice (Clipped Double Q-learning) and the value-function updates are delayed to improve stability.

Another approach applying SDRL to a CMDP energy system problem is presented by Li and He [25], solving the problem of Optimal Operation of Distribution Networks (OODN). This method uses the CPO algorithm previously presented in [4], which exploits a recently found connection of the difference in any return of two policies to an average divergence between these policies. Using this to get an upper and lower bound for the difference in policy performance between two policies, CPO modifies the PPO algorithm to introduce worst case surrogate functions for the objective function and the constraints, that can be easily evaluated from policy samples, to establish a trust region method that enforces an upper tolerance bound on constraint violations. This method can guarantee both prevention of constraint violations up to a small tolerance value as well as monotonic performance improvements, but is not very sample efficient and on-policy, because it is based on PPO.

4 Methodology

This chapter first describes and justifies the general approach that is followed in this thesis. The OPF simulator environments are described and the algorithms and frameworks are selected that are used to conduct experiments and collect data on the research questions. Finally, the strategies and algorithms used are explained in detail and the CTD3 algorithm is introduced.

4.1 General approach

Ensuring constraints in DRL approaches to CMDPs with added fixed linear penalties can come with the problem of having a trade-off between constraint satisfaction and policy optimality. Wang et al. [3] makes this clear by stating: "if one simply augments the reward with the product of a fixed penalty factor and constraint violation, then the learned policy will be either too conservative or infeasible". This problem of fixed penalty factors is also mentioned explicitly as a trade-off between reward and constraint cost in section 8.3 by Achiam et al. [4].

This problem is therefore also to be expected with the summation method mentioned in Chapter 3 that is used in many of the previous DRL approaches to OPF but not for the replacement method used in the other approaches as explained later. Research Question 1 of this thesis is, how relevant this trade-off is within the OPF problem. To answer this and to test the difficulty of finding the best penalty factor for every new OPF formulation, this thesis first measures the relevance of this trade-off by systematically comparing the objective and constraint performances of a variety of different penalty factors for the summation method. A state-of-the-art DRL algorithm is set up for two simulated OPF environments with different complexity and both the replacement method as well as the summation method from the DRL OPF literature are implemented. This makes the performance of the two penalty methods directly comparable within the OPF problem, which has not yet been done within the DRL OPF literature. After investigating the relevance of the trade-off for the summation method, the average performance of the replacement method is measured and the penalty methods are compared in performance, to answer Research Question 2

and test if the replacement method is a good trade-off free alternative to the summation method.

Afterwards, Research Question 3 is answered, investigating if SRL strategies are suitable alternatives to the fixed penalty methods for DRL approaches to the OPF problem. Therefore, a suitable and promising SDRL strategy is selected from the literature, applied to the previously used DRL agent and then tested on the different OPF simulators. Using the same base DRL algorithm, the objective performance and constraint satisfaction of the SRL approach are directly comparable to the fixed penalty methods.

Finally the results of the conducted experiments are analysed and conclusions towards the research questions are made. If the supposed trade-off of the summation method was shown in the first experiment, it is examined if the replacement method or the SDRL approach have improved or solved this problem in any way. If there has been no significant trade-off in the first place, it is analysed if the replacement method or SDRL approach has any other benefits or drawbacks compared to the summation method, for example improved policy optimality or if they can provide any guarantees for constraint satisfaction.

4.2 Selection of the algorithms

In this section, the algorithms used in this thesis are selected. To make the results of the different methods directly comparable, the fixed penalty methods and the SRL method have to be implemented into a common base DRL algorithm. Because implementing a full DRL algorithm is beyond the scope of this bachelor thesis, a selection of previously implemented DRL algorithms is at hand: TD3, which has many similarities to SAC, and Advantage Actor-Critc (A2C), which is a predecessor of PPO. The most promising SDRL algorithms for this thesis are CSAC and CPO as they aim to solve the trade-off and can provide guarantees for constraint satisfaction. CSAC is based on SAC and CPO is based on PPO, so none of the possible base DRL agents are at hand. Therefore the alternatives are, to either transfer the SRL strategy from CSAC or CPO to an algorithm at hand or to modify one of the algorithms at hand to an implementation of SAC or PPO.

All experiments and training processes of this thesis are done on a Desktop PC with limited computation power. To still be able to collect enough data for reliable conclusions in the limited time span of this thesis, it is necessary to use an algorithm with a good sample efficiency, that is able to learn nearly optimal behaviour within little amount of training time. Because the power system simulators are slow on the Desktop PC as well, an off-policy algorithm would also be preferable, to be able to let the agent learn from the collected past interactions.

PPO is an on-policy algorithm and less sample efficient than SAC and TD3, which are off-policy, so CPO inherits this properties from PPO. The surrogate objective used in CPO can not be easily applied to other algorithms than PPO, while the method of lagrange multipliers used in CSAC is independent of the base algorithm and therefore more easily transferable to one of the algorithms at hand. Therefore, the langrange relaxation approach used in the CSAC algorithm is the chosen SRL strategy used in this work. This choice is also expected to lead to higher performance, because CSAC has clearly outperformed CPO in the Volt-VAR problem in both constraint satisfaction and policy optimality. [3]

As mentioned earlier, a working TD3 implementation is at hand which has many similarities to SAC. There is no TD3 based SRL approach to energy systems yet, so this research gap is a great opportunity to transfer the SRL strategy used by CSAC to the TD3 algorithm, introducing a Constrained TD3 (CTD3) algorithm as a SRL variant of TD3. This approach is followed in this thesis, so TD3 is selected as the base DRL algorithm used in the experiments.

4.3 OPF Simulator Environments

The algorithms are tested in two different power system simulator environments of different complexity. The less complex simulates a reactive power market base case with the objective of minimizing reactive power costs and minimizing loss costs. The continuous actions in this simulator environment are the reactive power settings of all generators. The agent can observe the active and reactive power of the loads and the active power of the generators, as well as the prices for reactive power of the generators. For the constraints, the first simulator has limits for the voltage band, line/trafo load, min/max reactive power and does not allow reactive power to flow over the slack bus.

The second simulator is of higher complexity and represents an economic dispatch problem, with the objective of minimizing active power costs and minimizing loss costs. In this simulator, the actions are to set the active power of all generators and the agent can observe the active and reactive power of all loads and the active power prices of all generators. The constraints are limits for voltage band, line/trafo load and min/max active power limits.

The original rewards r_t of the simulation environments at timestep t come in vector form, with the first reward for the objective performance $f(x_t)$ of the current state x_t and multiple other rewards for each constraint violation $c_{t,k}(x_t)$, $k \in \{1, ..., n-1\}$.

$$r_t = \begin{pmatrix} f(x_t) \\ c_{t,1}(x_t) \\ \vdots \\ c_{t,n-1}(x_t) \end{pmatrix} \tag{4.1}$$

The rewards are predominantly negative, because the objective is a cost function and the constraint rewards are either zero or the negative amount of violation. For the penalty methods, the reward vector is transformed into a scalar reward using the penalty function of the currently investigated method. For the SRL method, the full reward vector is fed into the agent to have both the optimality and constraint information at hand.

4.4 Implementation of the penalty methods

To measure the relevance of a trade-off between objective optimality and constraint satisfaction as a problem of the summation method and to compare it with the replacement method, the two different linear penalty approaches are implemented. The summation method has the penalty function p_{sum} and the replacement method has the penalty function p_{repl}. Both methods are transforming a reward vector $r_j,\ j \in \{1, ..., n\}$ into a scalar reward, using a set of constraint penalty coefficients $\psi_k,\ k \in \{1, ..., n-1\}$ and for the summation method the common penalty factor ϕ .

$$p_{sum}(r) = r_1 + \phi \cdot \sum_{i=2}^{n} \psi_{i-1} \cdot r_i \tag{4.2}$$

$$p_{repl}(r) = \begin{cases} \sum_{i=2}^{n} \psi_{i-1} \cdot r_i & \text{if violations occurred} \\ 1000 + r_1 & \text{otherwise} \end{cases} \tag{4.3}$$

The summation method weights each constraint violation with a linear penalty, here the product of penalty factor and penalty coefficient $\phi \cdot \psi_k$, such that constraint violations reduce the overall reward of the agent. The replacement method has different rewards for different states of operation: If no violations occur, a high value is added to the objective reward, with 1000 being the value used in all previous approaches. If violations occur, the reward drops to a small value that is a weighted sum of the constraint violations. It is worth noting, that for the replacement method, constraint satisfaction is always more

important than objective performance, because with violated constraints the reward is of magnitudes smaller and not representing the objective anymore.

For the summation method, the penalty coefficients are balancing the importance of the different constraints to represent all constraint violations with a single value, which is then scaled by the penalty factor. This is a formal modification to the original method used in the literature and is introduced to seperate two problems of different relevance: the trade-off between constraint violations and policy optimality controlled by ϕ, which is of high interest for this thesis, and the weighting of the different constraints against each other. The ladder is less relevant to this thesis, as weighting the importance of different constraints against each other is not the research goal of this thesis and requires more in depth knowledge about the energy system, which is beyond the scope of this bachelor thesis. The coefficients can therefore be set by trial and error to result in penalized values of similar magnitude, which represents equal importance.

It is highly expected, that with the replacement method, no trade-off exists at all, because the constraint violation penalty factor is never effecting the optimality reward, there is no weighting between constraint satisfaction and optimality and the constraint satisfaction always has to be ensured first. Even with very low penalty factors the reward drops in magnitudes from a high objective value around 1000 to a low violation value below 0, which is assumed to be leading to very similar behaviour as using the summation method with high penalty factors. Therefore the trade-off is only investigated for the summation method, but after finding the best penalty factor, the replacement method is tested and compared with the summation method.

4.5 Implementation of CTD3

To transfer the SRL strategy used in CSAC from SAC to TD3, the differences between SAC and TD3 have to be considered. The main difference is, that SAC uses a stochastic policy while TD3 on the other hand uses a deterministic one. Another difference is the entropy maximization SAC proposes, adding an entropy term to the objective. The original SAC algorithm used in CSAC also learns an additional state value function which is left out by more recent versions of the SAC algorithm and not used with TD3. SAC and TD3 have in common, that they both use clipped double Q-Learning and delayed target updates.

4.5.1 Constrained SAC

In the following, the CSAC algorithm is explained in detail, to be able to transfer its inner workings to the TD3 algorithm.

CSAC uses a lagrange relaxation to learn a lagrange multiplier, which is used like a penalty factor in the summation method, linearly penalizing constraint violations in the relaxed Lagrangian objective. CSAC learns a Q-function for the Lagrangian reward and another Q-function for the constraint violation reward. The lagrange multiplier is then updated towards the minimal penalty factor necessary to reach constraint satisfaction within a given violation tolerance.

For SAC, V_h^π is the value function for the objective function of the CMDP, with the entropy maximization term $H(\pi(\cdot, s_t))$ from SAC added.

$$V_h^\pi(s) = \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{T}\gamma^t(R_t + \alpha H(\pi(\cdot, s_t))) \mid s_0 = s\right] \tag{4.4}$$

Within the maximum entropy framework of SAC, the optimal policy for the CMDP can be obtained by solving the following constrained optimization problem. D is the buffer of historical operation data and $V^{c,\pi}$ is the value function for the constraint violation R^c with $\overline{V^c}$ being an upper limit to a tolerated value of constraint violation.

$$\pi^* = \max_\pi \mathop{\mathbb{E}}_{s\sim D}[\, V_h^\pi(s)\,], \qquad s.t \qquad \mathop{\mathbb{E}}_{s\sim D}[\, V^{c,\pi}(s)\,] \le \overline{V^c} \tag{4.5}$$

CSAC uses a lagrange relaxation to transform the constrained optimization problem into an unconstrained one, by maximization of the relaxed Lagrangian objective $\mathcal{L}$ using a lagrange multiplier λ to penalize constraint violations. $V_h^{l,\pi}$ is the value function associated with the Lagrangian objective.

$$V_h^{l,\pi}(s) = \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{T}\gamma^t\,(\,R_t - \lambda R_t^c + \alpha H(\pi(\cdot, s_t))\,) \mid s_0 = s\right] \tag{4.6}$$

$$Q_h^{l,\pi}(s,a) = \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\, R_0 + \gamma V_h^{l,\pi}(s_1) \mid s_0 = s,\, a_0 = a\,\right] \tag{4.7}$$

$$\begin{aligned}\mathcal{L}(\pi,\lambda) &= \mathop{\mathbb{E}}_{s\sim D}[\, V_h^\pi(s)\,] + \lambda\left(\,\overline{V^c} - \mathop{\mathbb{E}}_{s\sim D}[\, V^{c,\pi}(s)\,]\,\right)\\ &= \mathop{\mathbb{E}}_{s\sim D}\left[\, V_h^{l,\pi}(s)\,\right] + \lambda\overline{V^c}\end{aligned} \tag{4.8}$$

With $\overline{V^c}$ being a constant, maximizing the Lagrangian means maximizing the Value function associated with the Lagrangian.

$$\pi^* = \max_{\pi} \mathcal{L}(\pi, \lambda) = \max_{\pi} \mathop{\mathbb{E}}_{s \sim D} \left[\; V_h^{l,\pi}(s) \; \right] \tag{4.9}$$

With the method of multipliers, the lagrange multiplier λ can be updated iteratively, with δ_λ being the learning rate of the update step. At k-th iteration, the optimal policy π^k is obtained for the current λ^k by maximizing $\mathcal{L}(\cdot, \lambda^k)$.

$$\lambda^{k+1} = \left[\; \lambda^k + \delta_\lambda \left(\mathop{\mathbb{E}}_{s \sim D} \left[\; V^{c,\pi^k}(s) \; \right] - \overline{V^c} \right) \; \right]^+ \tag{4.10}$$

In CSAC the lagrange multiplier and the policy and value networks are updated concurrently in the update step of the algorithm. With finite episodes, convergence can be proven with lambda update steps smaller than the learning rate of the policy network.

CSAC approximates $V_h^{l,\pi}$, $Q_h^{l,\pi}$, $V^{c,\pi}$ and $Q^{c,\pi}$ with neural networks using clipped double Q-Learning for both Q-functions. The Q-functions are updated using the current value functions and the value functions are updated using the current Q-functions.

4.5.2 Constrained TD3

With TD3 having a deterministic policy, the entropy maximization term can be removed from the value functions, resulting in simpler value functions for $V^{l,\pi}$ and $Q^{l,\pi}$.

$$V^{l,\pi}(s) = \mathop{\mathbb{E}}_{\tau \sim \pi} \left[\sum_{t=0}^{T} \gamma^t \left(\; R_t - \lambda R_t^c \; \right) \;\middle|\; s_0 = s \right] \tag{4.11}$$

$$Q^{l,\pi}(s,a) = \mathop{\mathbb{E}}_{\tau \sim \pi} \left[\; R_0 + \gamma V^{l,\pi}(s_1) \;\middle|\; s_0 = s,\, a_0 = a \; \right] \tag{4.12}$$

Because TD3 uses a deterministic policy and the state-values can be easily obtained from the Q-values, CTD3 only learns the Q-functions $Q^{l,\pi}$ and $Q^{c,\pi}$ with both clipped double Q-learning and uses them to obtain the value functions $V^{l,\pi}$ and $V^{c,\pi}$, using the average Q-value of the current learned on experience batch B with actions chosen with the current policy π.

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{\tau \sim \pi} \left[\; Q^{\pi}(s_0, \pi(s_0)) \;\middle|\; s_0 = s \; \right] \tag{4.13}$$

$$\approx \frac{1}{|B|} \sum_{B} Q^{\pi}(s_0, \pi(s_0)) \tag{4.14}$$

The core modifications of CTD3, compared to the base TD3 algorithm, are first the replacement of the reward R_t at timestep t, the TD3 Q-function is normally learned on, by the relaxed lagrange reward R_t^l and second the learning of another Q-function on the constraint violation reward R_t^c. Because in the simulators used in this work only one of the constraints is violated often, the CTD3 algorithm is simplified to use a single reward for all constraint violations, calculated as the weighted sum of the violation rewards from the reward vector $r_{t,i},\ i \in \{1, ..., n\}$. The weights ψ_{i-1} are analogous to the penalty coefficients in the penalty methods and are used to scale violations of the different constraint to a similar magnitude. For more complex environments, it is also possible to train the Q-function on all constraint rewards or even train an own Q-function and lagrange parameter for every constraint reward, if this simplification is not suitable. However, in this thesis a more complex architecture would make training even more time consuming, because training is done on a Desktop PC with limited computation power, therefore the simplification. The following shows how the reward vector $r_{t,i},\ i \in \{1, ..., n\}$ from the simulator is transformed into the CTD3 rewards.

$$\begin{aligned} R_t &= r_{t,1} \\ R_t^c &= \sum_{i=2}^{n} \psi_{i-1} r_{t,i} \\ R_t^l &= R_t + \lambda R_t^c \end{aligned} \tag{4.15}$$

All Q-functions are learned by minimizing their mean squared error, both using the clipped double Q-Learning trick and both using delayed target updates and soft target updates as used in base TD3.

Using clipped double Q-learning, both Q-functions are represented with two DNNs each, resulting in two sets of Q-networks $Q_{\phi_1} = \{Q_{\phi_1^c}, Q_{\phi_1^l}\}$ and $Q_{\phi_2} = \{Q_{\phi_2^c}, Q_{\phi_2^l}\}$, parameterized by ϕ^l for the lagrange objective and by ϕ^c for the constraints. The policy network π_θ, parameterized with θ, is updated with gradient ascend towards the policy $\hat{\pi}$ that maximizes the expected lagrange Q-values of the batch, as estimated by the current Q-networks.

$$\hat{\pi} = \arg\max_{\pi} \mathop{\mathbb{E}}_{\tau \sim \pi} \Big[Q_{\phi^l}(s_0, \pi(s_0)) \ \mid \ s_0 = s \Big] \tag{4.16}$$

Like with base TD3, the Q-value targets $\hat{Q}^l$ and $\hat{Q}^c$ are obtained by using the current policy network π_θ, for temporal difference learning on a random batch from the experience

buffer, which for CTD3 is extended to contain the whole reward vectors received from the environment.

$$\begin{aligned}\hat{Q}^l(s_t, a_t) &= R_t^l + Q_{\phi^l}(s_{t+1}, \pi_\theta(s_t)) \\ \hat{Q}^c(s_t, a_t) &= R_t^c + Q_{\phi^c}(s_{t+1}, \pi_\theta(s_t))\end{aligned} \tag{4.17}$$

The Q-networks as well as the policy network are learned using soft target updates to decrease variance and stabilize learning. Soft target updates manage a copy of every DNN, that is used instead of the actual networks and slowly updated towards the parameters ϕ of the trained network, using a weight $\rho \in (0, 1)$.

$$\phi_{targ} = (1 - \rho)\phi_{targ} + \rho\phi \tag{4.18}$$

The are some additional modifications made in CTD3 to stabilize the learning of the λ-parameter further, to reduce problems that are observed in test runs of the algorithm. The first modification is, to delay the start of λ-learning process to a learning step, where the Q-functions have already converged to a good approximation. For training runs with 25000 episodes this point of time is set 1500 episodes after Q-function learning started and for training runs with 50000 episodes λ-learning starts 3000 episodes after Q-function learning started. This avoids having extremely high λ-learning steps at the beginning of training, because the first estimated constraint Q-values can be extremely high, which leads to extremely high constraint state-values and therefore high λ-learning steps and far too high λ values. This is often not recoverable, because the high λ values result in high Lagrange Q-function losses and destructive policy updates, which again result in high constraint Q-value losses. Delaying λ-learning to start learning after the Q-functions have settled, solves this feedback-loop.

Another modification made to balance λ-learning is intended for constraints, that can reach very high violation values at start of λ-learning. The solution is, to clip the λ-learning gradient to a small maximal stepsize ϵ. This allows to set a higher λ-learning rate δ_λ necessary to reach high constraint satisfaction within short training runs, without having the λ-parameter grow out of bounds at start of λ-learning.

$$\lambda^{k+1} = \left[\lambda^k + CLIP\left(\delta_\lambda \left(\mathop{\mathbb{E}}_{s \sim D} \left[V^{c,\pi^k}(s) \right] - \overline{V^c} \right),\ -\epsilon,\ \epsilon \right) \right]^+ \tag{4.19}$$

5 Experimental setup

To answer the three research questions, three experiments are conducted. To answer Research Question 1, the first experiment investigates the relevance of the trade-off by testing the given TD3 DRL agent with the summation method using a variety of different penalty factors. If the trade-off is relevant, high penalty factors should lead to high constraint satisfaction at cost of low objective optimality and low penalty factors should achieve the highest objective optimality at cost of high amounts of constraint violations. The trade-off is only investigated for the summation method, because the experiment of trying many penalty factors is very time consuming and the replacement method is highly expected to have no trade-off at all, as explained earlier. The second experiment is set up to answer Research Question 2, investigating the potential of the replacement method as an alternative to the summation method. The performance regarding objective optimality and constraint satisfaction of the replacement method is compared to to the performance of the summation method to find the most effective commonly used constraint satisfaction strategy. To answer Research Question 3, the promising SDRL strategy of a learned lagrange multiplier is tested by implementing the proposed CTD3 algorithm by extending the previously used TD3 agent. In the third experiment the performance of CTD3 is compared to the fixed penalty methods, testing if SDRL methods can achieve more effective constraint satisfaction or guarantees for constraint satisfaction.

5.1 General setup

The experiments are conducted based on a TD3 algorithm previously implemented in python using pytorch [26]. The hyperparameters of the algorithm are set the same for all three conducted experiments. Both the actor network as well as the critic networks have two hidden layers with 500 neurons each. The optimizer of the networks is set to *Adam* [27] and the networks losses are mean squared error losses. For the actor, the learning rate is 0.0001 and for the critic it is 0.0001.

5.2 Experiment 1: Investigation of the trade-off for the summation method

For the summation method, an experiment is set up, that systematically runs $s \in \{1, ..., M\}$ sub experiments, such that the penalty factor ϕ_s doubles with every experiment, starting with start value ω. For every sub experiment s, five training sessions are conducted then, to provide enough reward data for each penalty factor. An exponentially increasing reward is chosen, because this allows comparing the effects of a wide range of penalty factors in a very efficient way.

$$\phi_s = \omega \cdot 2^{s-1} \tag{5.1}$$

The experiment is conducted on two OPF simulators of differing complexity. With $\omega = 0.04$ and $M = 10$ the penalty factor covers values between 0.04 and 20.48. With five training session per experiment, there is enough data to draw conclusions, without having a time expenditure too high for the scope of this thesis. On the simplest simulator 5000 training steps are used, 10000 on the more difficult simulator and 20000 on the most difficult one, because the more complex the simulator, the more time the agent needs to learn. The full reward history of each training session is saved to disk in CSV-format, using the original reward vectors from the simulation environment, to have a common metric between all experiments. The results are analysed by first averaging each rewards over the last 10% of training steps to get a final average performance for every reward after each training session. The final performances of the five training sessions of each penalty factor are then again averaged, to get the average final rewards for every penalty factor.

If the supposed trade-off is relevant, the optimality reward should get more negative (higher cost) with an increasing penalty factor and the constraint violation value should get closer to zero.

5.3 Experiment 2: Comparison between the summation and the replacement method

Because the mentioned trade-off is not to be expected for the replacement method, it could be a good alternative to the summation method, if the policy optimality is comparably high. But this is not very likely, as with the replacement method, the constraint satisfaction is always more important to the agent then the policy optimality, so it is expected to behave like a very high penalty factor. On the other hand, the policy gradient is never distorted by

the penalties, which might allow the replacement method to converge to a more optimal policy than the summation method. To investigate this, an experiment is set up to test the average performance of the replacement method both OPF simulator environments. For every simulator, five training sessions are conducted with 10000 steps for the most simple simulator, 25000 steps for the more complex simulator and 50000 steps for the most complex one. The higher number of training steps is chosen, because in contrast to the 150 training sessions conducted in Experiment 1 ($10 \cdot 5 = 50$ for each simulator), Experiment 1 only conducts 15 training sessions so time is less relevant. Furthermore, the value function approximations need some time to reach the high values over 1000, so the replacement method is expected to converge slower. Because the results with more training steps are not directly comparable with the results from Experiment 1, Experiment 2 is conducted again for the summation method but with a penalty factor that had the best trade-off in Experiment 1. For both methods, the rewards are saved to disk in CSV format and the average training rewards for both objective and constraints are averaged over all 5 sessions and over the last 10% of training steps of each session to get an approximation of the final performance.

5.4 Experiment 3: Constrained TD3

To test the objective and constraint performances of CTD3, an experiment comparable to Experiment 2 is conducted that trains the CTD3 agent five times for each simulation environment. The results have to be directly comparable to Experiment 2, so hyperparamters and training duration remain unchanged. The additional CTD3 hyperparameters are set as following: The lagrange multiplier λ is initialized to zero, as recommended in the CSAC paper [3]. For the lower complexity simulator, the lagrange multiplier learning rate δ_λ is set to 0.005 and the upper limit for the lagrange multiplier update step ϵ is set to 0.002. For the higher complexity simulator, δ_λ is set to 0.001 and ϵ is set to 0.0001. This hyperparameters are found by trial an error in test runs to result in a slowly increasing λ-parameter that still increases fast enough to reach convergence to the best penalty factor during the 25000 and 50000 training steps. The tolerated value of constraint violation $\overline{V^c}$ is set to 0.05 for both simulators, lower then the average violation of the best results of the penalty methods but not too small as this can make the lagrange multiplier grow out of bounds, resulting in too conservative policies. The learning of the lagrange multiplier is delayed to start after 3000 training steps, because this allows the Q-function networks to settle for a good approximation before the lagrange learning starts. This also allows the agent to first learn the objective and then slowly begin to consider the constraints as well.

Again, the rewards are saved to disk in CSV format and the final performance for both objective and constraint satisfaction is averaged over all ten sessions and over the last 10% of training steps. Afterwards the average total performance of the CTD3 algorithm regarding objective and constraint satisfaction can be compared to the penalty methods and it can be discussed, how and if CTD3 has solved the trade-off problem mentioned earlier and if CTD3 can provide guarantees for constraint satisfaction.

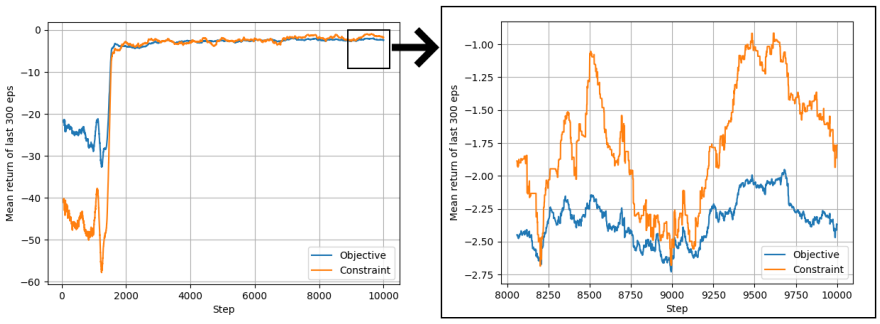
6 Results

The following sections describe the results of the experiments in detail. For every experiment, first some important observations from the training processes of the used method are provided, followed by the final result data of each experiment.

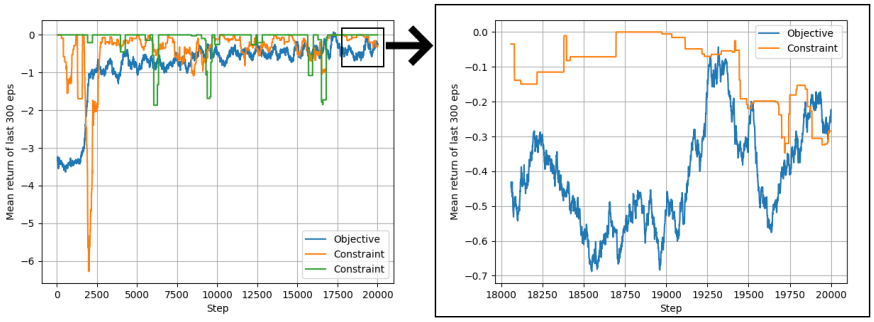
6.1 Experiment 1: Investigation of the trade-off for the summation method

6.1.1 Training process

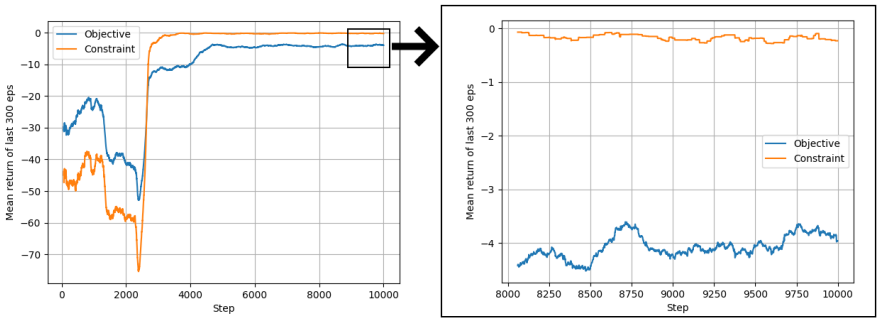
For both simulator environments 50 training session are run, resulting in a total of 100 training sessions. On the used Desktop PC, a low complexity simulator training session with 10000 training steps takes on average 10 minutes and a high complexity simulator training session with 20000 training steps takes on average 20 minutes, so the full training process of Experiment 1 has a time expenditure of roughly $1500/60 = 25$ hours. Figure 6.1 show the performance progression of the objective reward and the constraint violations scaled by the penalty coefficients of four training sessions, having both the lowest and highest penalty factors of 0.04 and 20.48 on both simulators. The original reward data has very high variance, so the graphs show the mean rewards of the last 300 episodes. At the first 1000 training steps, training has not yet started and the agent is filling the experience buffer, so the performance is very low and no improvement is observable. Afterwards, training begins and a short drop of performance happens and then the curves quickly move to a high level of performance that is then slowly improved upon. With the low penalty factor, the constraint violations occur more often but the average objective reward reaches higher levels than with the high penalty factor. The high penalty factor on the other hand results in minimal amounts of constraint violations but with worse objective performance. In the high complexity simulator the highest penalty factor even leads to both extremes, perfect constraint satisfaction but with the objective performance staying on the initial level.



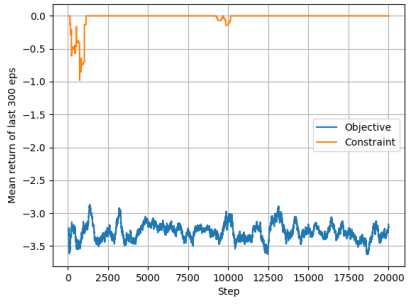
(a) Penalty factor 0.04 on lower complexity Simulator



(b) Penalty factor 0.04 on higher complexity Simulator



(c) Penalty factor 20.48 on lower complexity Simulator



(d) Penalty factor 20.48 on higher complexity Simulator

Figure 6.1: Selection of training sessions of Experiment 1: Average objective and constraint rewards of a gliding window of the last 300 training steps using the summation method with a low penalty of 0.04 and a high penalty of 20.48 on two simulators of different complexity.

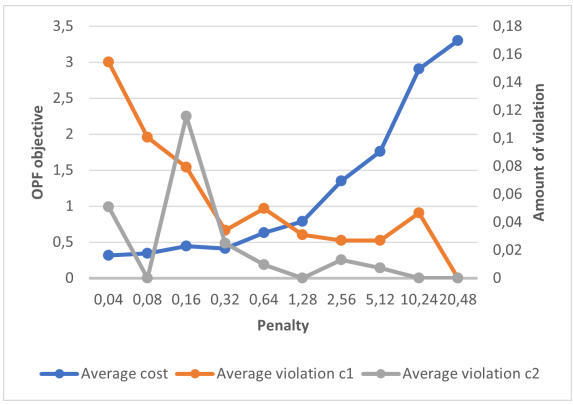
6.1.2 Final results

The final results of Experiment 1 show a significant trade-off between objective optimality and constraint satisfaction. Figure 6.2 shows the average final 10% training objective costs and constraint violations of the summation method over a range of ten different penalty factors, exponentially growing from 0.04 to 20.48. The figure shows the costs and violations as positive values in contrast to the negative reward value that are used in the results of the other experiments.

The higher complexity simulator shows the most significant trade-off. With the lowest penalty factor the best average objective cost 0.31 is reached but the average constraint violations are very high at 0.15. With increasing penalty factor, the objective performance gradually worsens and the constraint satisfaction improves. The highest penalty factor then comes with perfect constraint satisfaction with 0 average violations, but has an objective cost of 3.3, ten times worse than with the low penalty.

On the lower complexity simulator the trade-off is less significant. The objective cost stays between 2.0 and 2.5 while the constaint violations improve from 1.66 to 0.24 until the penalty factor reaches 5.12. Afterwards, the objective costs drop to 3.62 as the constraint violations stay roughly the same around 0.24.

(a) Lower complexity OPF Simulator



(b) Higher complexity OPF Simulator

Figure 6.2: Final results of Experiment 1: Penalty factor trade-off between average amount of constraint violation and average OPF objective cost, tested on two simulators of different complexity.

6.2 Experiment 2: Comparison between the summation and the replacement method

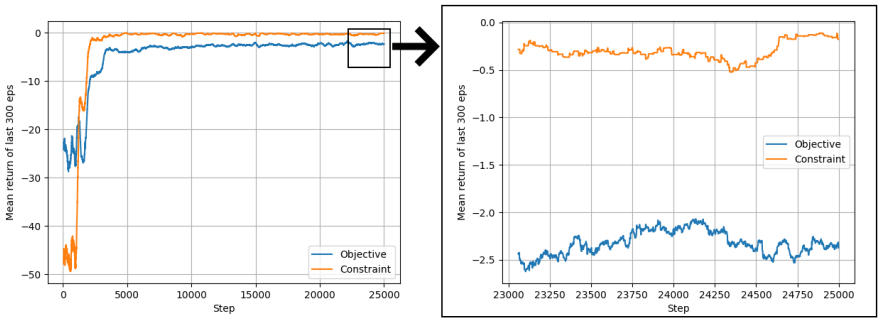
To measure and compare the average performance of both methods, five training sessions are run for both the summation method with the best penalty factor as well as the replacement method with 25000 steps on the lower complexity simulator and 50000 steps on the higher complexity simulator, with a total time expenditure of around 12.5 hours. The penalty factor of the summation method is set to 5.12 for the lower complexity simulator and to 1.28 for the higher complexity simulator, as they have the most balanced compromise in the trade-off between optimality and constraints, according to the results of Experiment 1. The performance metric in this section is the average final training reward, so in contrast to the results of Experiment 1, the performance values shown here are always negative.

6.2.1 Training process

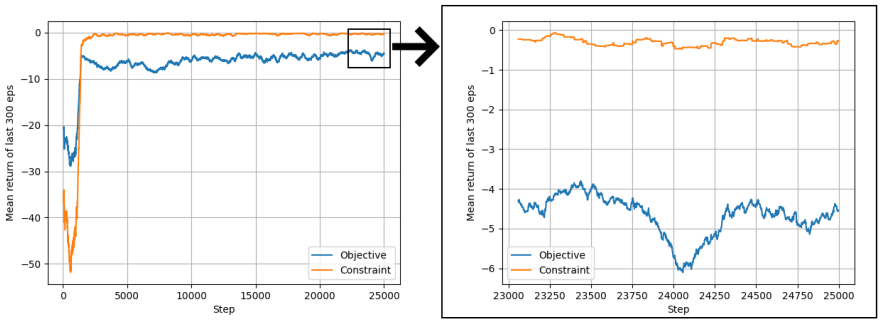
In the training processes shown in Figure 6.3, some differences between the summation method and the replacement method are observable. With the summation method, the constraints come to a high performance of -0.3 first, shortly followed by a moderate objective performance between -3 and -4 that slowly improves to a high performance around -2.3. With the replacement method on the other hand, constraint and objective performance start to rise simultaneously but shortly followed by a drop in objective performance to a low value around -7. The objective performance never reaches a very high level but slowly improves to a slightly better low value of -5, while the constraint performance reaches high values on the same level as the summation method around -0.3.

6.2.2 Final results

The results Experiment 2 are presented in Table 6.1. The replacement method reaches far lower objective rewards than the summation method, with -3.83 compared to -2.33 for the lower complexity simulator and -3.32 compared to -0.66 for the higher complexity simulator. For the constraint satisfaction, both methods reach roughly the same average reward of -0.29 on the lower complexity simulator but different rewards on the higher complexity simulator, with the replacement method reaching -0.013 for the first constraint and 0.0 for the second, whereas the summation method reaches -0.016 for the first constraint and -0.0066 for the second.



(a) Summation method



(b) Replacement method

Figure 6.3: Selection of training sessions of Experiment 2: Average objective and constraint rewards of a gliding window of the last 300 training steps using the summation method with penalty of 5.12 and the replacement method on the lower complexity OPF simulator.

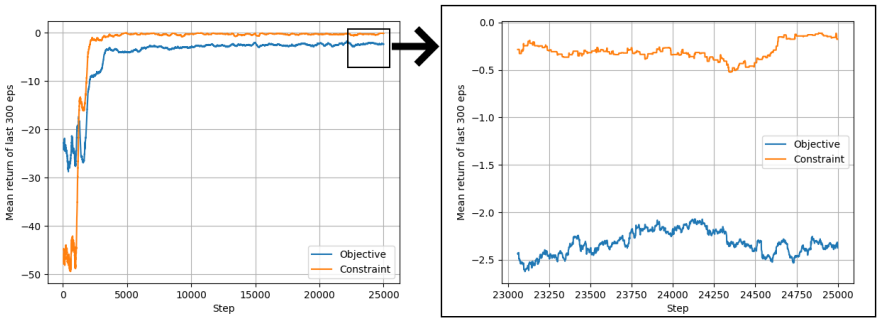
Simulator type	Low complexity		High complexity		
Reward	objective	constraint	objective	constraint 1	constraint 2
Summation method	−2.33170	−0.28746	−0.65634	−0.01627	−0.00660
Replacement method	−3.83137	−0.28978	−3.32034	−0.01258	0.0

Table 6.1: Experiment 2: Average final 10% of training rewards of the replacement method compared to the summation method with selected penalty factor, for both objective and constraint satisfaction. The experiment is run on two OPF simulators of different complexity with 25000 training steps for the low complexity simulator and 50000 training steps for the high complexity simulator

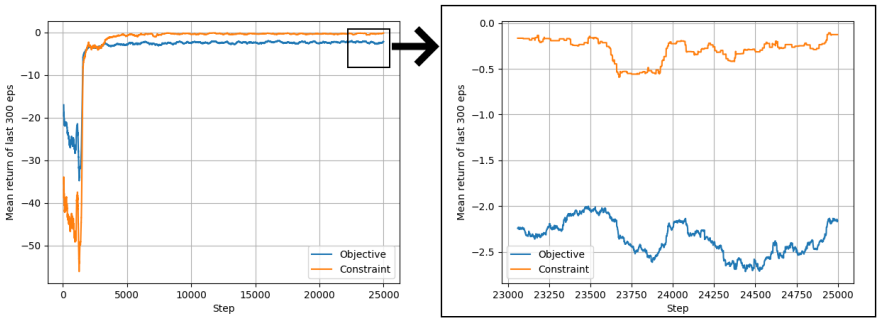
6.3 Experiment 3: Constrained TD3

The new CTD3 algorithm is tested on the low complexity simulator with 25000 training steps and on the high complexity simulator with 50000 training steps.

6.3.1 Training process



(a) Summation method



(b) CTD3 method

Figure 6.4: Selection of training sessions of Experiment 3: Average objective and constraint rewards of a gliding window of the last 300 training steps using the summation method with penalty of 5.12 and the CTD3 algorithm on the lower complexity OPF simulator.

The training process of CTD3 is shown in Figure 6.4. There is one central difference to the training process of the summation method: The constraint performance of the summation method reaches a high level right after start of training, while in the CTD3 algorithm the constraint performance first rises to a very low level of -3.5 but after the start of lambda learning, it jumps to a very high level of -0.4, close to the final level of -0.3. The objective performance on the other hand is not delayed as in the summation method and reaches a high level around -2.5 right after training started.

Although the lagrange multiplier development is not recorded as result data, the training console outputs the current lagrange multiplier and the current expected constraint violation from the additional Q-functions. It is observable, that the Q-function constraint violation expectations are far smaller than the actually happening constraint violations and that the lagrange multiplier stops growing before the accepted violation tolerance is reached. The mean error of the constraint Q-functions is between 0.5 and 1.0.

6.3.2 Final results

In Table 6.2, the final results of Experiment 3 are presented. On the lower complexity simulator, CTD3 achieves slightly higher average final objective rewards of -2.28 compared to the summation method with -2.33. It also comes with slightly higher constraint satisfaction of -0.27 compared to -0.29 of the summation method. On the higher complexity simulator, the objective performance is lower, reaching -0.74 with CTD3 and -0.66 with the summation method, but the constraint satisfaction is higher, achieving -0.011 compared to -0.016 for the first constraint and -0.0003 compared to -0.0066 for the second. Except for the constraint 2 violations, CTD3 outperforms the replacement method in all rewards on both simulators.

Simulator type	Low complexity		High complexity		
Reward	objective	constraint	objective	constraint 1	constraint 2
Summation method	-2.33170	-0.28746	-0.65634	-0.01627	-0.00660
Replacement method	$-3,83137$	-0.28978	-3.32034	-0.01258	0.0
CTD3 algorithm	-2.28032	-0.27302	-0.74297	-0.01106	-0.00028

Table 6.2: Experiment 3: Average final 10% of training rewards of the CTD3 algorithm compared to the summation method with selected penalty factor and the replacement method, for both objective and constraint satisfaction. The experiment is run on two OPF simulators of different complexity with 25000 training steps for the low complexity simulator and 50000 training steps for the high complexity simulator

7 Discussion

In this chapter, the implications of the experimental results are analysed to answer the research questions of this thesis, investigating if the research goals are reached. Previously made expectations are compared with the results and possible explanations for some of the results are provided. The approaches taken in this thesis are reflected upon and its advantages and disadvantages are discussed. Finally the limitations of this work are explained as well as unanswered questions and newly opened research gaps for following work.

The research goal of this thesis is to find more effective constraint satisfaction strategies for DRL approaches to the OPF problem. As defined in Chapter 1, this thesis aims to answer the three research questions:

Research Question 1:
How relevant is a trade-off between objective performance and constraint satisfaction for the fixed penalty DRL approaches to the OPF problem?

Research Question 2:
Which advantages and disadvantages regarding effective constraint satisfaction does the replacement method have, compared to the summation method?

Research Question 3:
Which advantages and disadvantages regarding effective constraint satisfaction do SDRL approaches have for the OPF problem, compared to the fixed penalty methods?

7.1 Literature and expectations

Some theoretical results can be deduced from the literature in Chapter 3, which are important for the research questions as they provide a basis for important expectations. First, most of the previous approaches use fixed penalty functions to achieve constraint satisfaction, only one approach used a lagrange relaxation. Second, two SRL papers suggest that fixed penalty methods come with the drawback of a trade-off between objective

and constraints [25], [4]. It is highly expected, that the same problem is also relevant to the previous DRL approaches to OPF. Third, both mentioned SRL papers provide SDRL algorithms for CMDP, CSAC and CPO, that are designed to solve or avoid the trade-off, showing higher overall performance compared to fixed penalty methods in other CMDPs. Therefore, the CSAC and the CPO algorithms are highly expected to also achieve more effective constraint satisfaction for the OPF problem, compared to the fixed penalty methods.

7.2 Interpretation of the experimental results

The first two experiments clearly demonstrate the drawbacks of the constraint satisfaction strategies of previous approaches. Confirming the previously made expectations, the trade-off between policy optimality and constraint satisfaction is demonstrated with both simulator environments when using the summation method, which makes finding the best penalty factor a time consuming optimization problem in itself.

As seen in Experiment 1, high penalty factors correlate with high levels of constraint satisfaction but low objective performances, while low penalty factors correlate with low constraint satisfaction and high objective performance. Because every penalty factor is tested 5 times and the penalty factor is the only parameter that changes between the training sessions, it is safe to interpret this correlation as a causal. This implies, that high overall performance is only possible if the penalty factor is set to optimally balance the trade-off between objective and constraints, which is not trivial to find. Therefore, the answer to Research Question 1 is, that the trade-off is highly relevant for applications of the summation method to OPF problems.

Experiment 2 shows, that although the replacement method avoids the problem of having to find the right penalty factor, it is not a very effective constraint satisfaction strategy either, because it behaves just like the summation method with a very high penalty factor, having high constraint satisfaction at cost of very low objective performance. Therefore, the replacement method has the advantages of high constraint satisfaction and that it avoids finding the best penalty factor, but the disadvantages of having far lower objective optimality and less control over the trade-off, which answers Research Question 2.

The most likely explanation for the trade-off is the statement made by Wang et al. [3] that high penalties lead to policies that are too conservative, avoiding to choose actions that get close to constraint violations, although they may lead to the highest objective rewards within the constraints. Furthermore, the global optimum of the objective function probably does not lie within the constraints, which can make it profitable for the agent to violate constraints if low penalty factors are used. This also means, that the best objective

performances found in the experiments are probably only achievable with high constraint violations, whereas the optimal OPF solutions within the constraints can have far lower performances.

Experiment 3 shows, that the proposed CTD3 algorithm can solve OPF with slightly more effective constraint satisfaction than the summation and the replacement method. Compared to the best penalty factor summation method, it reaches both higher objective performance and slightly higher constraint satisfaction on the lower complexity simulator and higher constraint satisfaction and still comparably high objective performance on the high complexity simulator. A comparable performance is achieved on both simulators, because the lagrange multiplier is updated to converge towards the most optimal penalty factor, that can guarantee constraint satisfaction within a given violation tolerance. The main advantage of CTD3 therefore is, that it can automatically find the optimal penalty factor, making it unnecessary to find it by hand for every new OPF problem. The lower objective performance in the high complexity simulator indicates that the trade-off might still be relevant with CTD3. An explanation for this is, that the summation method is only able to reach higher objective performance, because it has higher constraint violations and the higher objective rewards lie outside of the constraints. Therefore it is difficult to know the highest objective performance that is reachable within the constraints and it is still possible that the objective performance of CTD3 is in fact near optimal within the constraints. However, the relevance of the trade-off is measurably reduced with CTD3: On the lower complexity simulator, CTD3 achieves higher objective performance with at least the same level of constraint violations as the summation method and on the high complexity simulator CTD3 achieves near perfect constraint satisfaction comparable to the replacement method with still very high objective performance. A possible explanation for this is, that the delayed start of the penalty learning leads to more optimal policies. In theory, a policy that is learning the objective first and then begins to learn constraint satisfaction, should reach higher objective performance faster than a policy that learns constraint satisfaction first, because it is easier to find the local optimum within the constraints after the global optimum of the objective function has already been found. The summation and replacement method both start learning constraints and objective simultaneously, so this is an important difference. The results of Experiment 3 answer Research Question 3: The CTD3 SDRL algorithm has the advantage of more effective constraint satisfaction, reaching both high levels of constraint satisfaction and objective performance without having to find the best penalty factor by hand. The disadvantage of CTD3 is, that the new λ-learning rate and constraint violation tolerance hyperparameters have to be tuned by hand to reach high constraint and objective performance.

With the results of the three experiments, the research questions are answered. The trade-off between constraint satisfaction and objective optimality is shown as highly relevant for the summation method. The replacement method is analysed as being too conservative so it is not a more effective constraint satisfaction strategy than the summation method. The potential of SRL for this problem is measurably demonstrated. CTD3 as a simplified TD3 version of CSAC is presented by transferring its SRL strategy to the CTD3 algorithm, and shown as a suitable approach to the OPF problem as it reaches both high constraint satisfaction and objective performance. The research goal of finding more effective constraint satisfaction strategies is therefore reached, because with CTD3 the trade-off is less relevant, so this is an example, how more effective constraint satisfaction is possible using SRL.

7.3 Limitations of this work

Although the research goals are reached and the results of the experiments are significant, it is important to note the limitations of this work. First of all, there are important technical limitations. The experiments have only be conducted on two energy simulators, so other simulators might lead to different results. Furthermore, the training of the neural networks is done on CPU using a desktop PC with limited computation power. Because the conducted experiments are very time consuming, the neural networks are not very large, having only two layers of 500 neurons each. To verify the findings of this thesis, the results have to be reproduced with more complex power simulators and more complex neural networks that can be trained on compute clusters.

For the CTD3 algorithm, there are some important limitations, too. First of all, the CTD3 algorithm is at its core a penalty method comparable to the summation method. The main differences are the automated learning of the penalty factor and the delayed learning of the constraints. Therefore, the trade-off problem might still be relevant to the CTD3 algorithm. Although CTD3 can provide limited guarantees for constraint satisfaction, it can not provide a general guarantee of finding an optimal policy. An important problem found in the experiments of the CTD3 algorithm is, that the Q-function estimators for the constraints have high losses, which lead to underestimated constraint violation Q-values being used in the lagrange update step. This results in the problem, that the penalty factor stops growing before the constraint tolerance is reached. This can be easily solved by setting the constraint tolerance to a smaller value, but it is not clear how small the value has to be set to achieve the intended constraint satisfaction guarantee. This problem might be related to the used small neural networks, that might not be able to represent the complexity of the constraints, so larger neural networks might fix this problem. Another reason for this problem might be the fact, that CTD3 simplifies CSAC by not learning the state-value

functions and approximating them with the Q-function. Therefore a reintroduction of the additional state-value function could solve this problem. From a theoretical standpoint, the CTD3 algorithm can provide guarantees for constraint satisfaction, if the constraint Q-function estimators are precise enough.

7.4 Open questions

Experiment 1 has shown, that the trade-off of the summation method is much more significant for the more complex OPF simulator than for the less complex simulator. This indicates that the significance of the trade-off is related to the complexity of the used OPF-formulation. Therefore, an interesting newly opened research question is, how the relevance of the trade-off is related to the complexity of the OPF formulation.

This thesis does not implement the original CSAC algorithm but a simplified TD3 version of it. Therefore, an important question is, if the original CSAC algorithm can provide even more effective constraint satisfaction for the OPF problem.

Another important question is, how CTD3 compares to the summation method, if every constraint has its own penalty factor, especially important in OPF formulations with a large number of easily violated constraints. The CTD3 algorithm could learn a lagrange multiplier for every constraint and therefore avoid a multi-dimensional trade-off, that is expected to be far more time consuming to solve by hand.

Unfortunately, the theoretical guarantees for constraint satisfaction of CTD3 are not realized in this thesis, because the Q-function network losses are too high. This could be easily solved with more computational power and larger neural networks. So the question arises, how the CTD3 algorithm scales compared to the summation and the replacement method.

It is also not clear, how CTD3 achieves the reduced relevance of the trade-off. A possible explanation is, that it makes a difference in which order multiple tasks are learned by a neural network, because the order determines the gradient descend/ascend trajectory of the network parameters through the optimization landscape. In case of OPF a policy network that has already found the global optimum of the objective function might be closer to the constrained local optimum than a policy network that is only capable of avoiding constraint violations. It is an open question if this is true but the answer probably has broad implications for other DL problems with multiple tasks as well.

8 Conclusion

After conducting three experiments, this thesis shows that more effective constraint satisfaction is necessary for DRL approaches to the OPF problem and that it can be improved with SRL strategies. In Experiment 1 and Experiment 2, the fixed penalty function methods used in most previous approaches are applied to the TD3 algorithm and tested on two different OPF simulators. The experiments show how both methods suffer from a trade-off between policy optimality and constraint satisfaction. In Experiment 1, the commonly used summation method is tested with a variety of different penalty factors, showing that finding the optimal penalty factor is difficult and that the method comes with either suboptimal policies or unacceptable amounts of constraint violations. As expected, high penalty factors lead to a policy that is too conservative and low penalty factors lead to unacceptable amounts of constraint violations. Experiment 2 shows how the same trade-off is inherent to the other commonly used replacement method, that can achieve high performance for constraint satisfaction but at cost of a highly suboptimal policy. The replacement methods behaves like the summation method with a very high penalty factor, with the summation method with a optimally chosen penalty factor being the most effective commonly used constraint satisfaction strategy. After demonstrating the need for more effective constraint satisfaction strategies, a first promising candidate is implemented. The lagrange multiplier SRL strategy used in the CSAC algorithm is transferred to the TD3 algorithm, proposing CTD3 as a novel SRL algorithm. In Experiment 3, the novel CTD3 algorithm is compared to the fixed penalty methods and achieves high performance of constraint satisfaction and policy optimality, suffering less from the mentioned trade-off. CTD3 is able to automatically find the best penalty factor, saving the time to systematically search for it for every new OPF problem. In theory, CTD3 can even provide a tolerance guarantee for constraint satisfaction, if the constraint Q-functions have very low losses. Unfortunately, this is not achieved in the implementation used in this work, because the constraint Q-function losses are too high, as the used neural networks are comparably small and the training runs are short to save computation time. Nevertheless, the research questions are answered and the research goal of finding more effective constraint satisfaction strategies for DRL approaches to OPF is reached.

List of Figures

List of Tables

Bibliography

[1] Frank, S., Steponavice, I., Rebennack, S.: Optimal power flow: a bibliographic survey i: Formulations and deterministic methods. Energy systems (Berlin. Periodical) **3**(3), 221–258 (2012)

[2] Hasan, F., Kargarian, A., Mohammadi, A.: A survey on applications of machine learning for optimal power flow. In: 2020 IEEE Texas Power and Energy Conference (TPEC), pp. 1–6 (2020). doi:10.1109/TPEC48276.2020.9042547

[3] Wang, W., Yu, N., Gao, Y., Shi, J.: Safe off-policy deep reinforcement learning algorithm for volt-var control in power distribution systems. IEEE Transactions on Smart Grid **11**(4), 3008–3018 (2020). doi:10.1109/TSG.2019.2962625

[4] Achiam, J., Held, D., Tamar, A., Abbeel, P.: Constrained Policy Optimization. arXiv (2017). doi:10.48550/ARXIV.1705.10528. https://arxiv.org/abs/1705.10528

[5] Mousavi, S.S., Schukat, M., Howley, E.: Deep reinforcement learning: An overview. In: Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016, pp. 426–440. Springer, (2017). doi:10.1007/978-3-319-56991-8_32. https://doi.org/10.1007%2F978-3-319-56991-8_32

[6] Silva, I.N.d., Spatti, D.H., Flauzino, R.A., Liboni, L.H.B., Alves, S.F.d.R.: Artificial Neural Networks - A Practical Course. Springer, Berlin, Heidelberg (2016)

[7] McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics **5**(4), 115–133 (1943). doi:10.1007/BF02478259

[8] Rosenblatt, F.: The perceptron – a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory (1957)

[9] Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review **65**(6), 386–408 (1958). doi:10.1037/h0042519

[10] Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Gordon, G., Dunson, D., Dudík, M. (eds.) Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 15, pp. 315–323. PMLR, Fort Lauderdale, FL, USA (2011). https://proceedings.mlr.press/v15/glorot11a.html

[11] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986). doi:10.1038/323533a0

[12] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT press, (2018)

[13] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning (2019). 1509.02971

[14] Fujimoto, S., van Hoof, H., Meger, D.: Addressing Function Approximation Error in Actor-Critic Methods (2018). 1802.09477

[15] Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor (2018). 1801.01290

[16] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal Policy Optimization Algorithms (2017). 1707.06347

[17] Schulman, J., Levine, S., Moritz, P., Jordan, M.I., Abbeel, P.: Trust Region Policy Optimization (2017). 1502.05477

[18] García, J., Fern, o Fernández: A comprehensive survey on safe reinforcement learning. Journal of Machine Learning Research **16**(42), 1437–1480 (2015)

[19] Cao, D., Hu, W., Xu, X., Wu, Q., Huang, Q., Chen, Z., Blaabjerg, F.: Deep reinforcement learning based approach for optimal power flow of distribution networks embedded with renewable energy and storage devices. Journal of Modern Power Systems and Clean Energy **9**(5), 1101–1110 (2021). doi:10.35833/MPCE.2020.000557

[20] Yan, Z., Xu, Y.: Real-time optimal power flow: A lagrangian based deep reinforcement learning approach. IEEE Transactions on Power Systems **35**(4), 3270–3273 (2020). doi:10.1109/TPWRS.2020.2987292

[21] Zhou, Y., Zhang, B., Xu, C., Lan, T., Diao, R., Shi, D., Wang, Z., Lee, W.-J.: A data-driven method for fast ac optimal power flow solutions via deep reinforcement learning. Journal of Modern Power Systems and Clean Energy **8**(6), 1128–1139 (2020). doi:10.35833/MPCE.2020.000522

[22] Woo, J.H., Wu, L., Park, J.-B., Roh, J.H.: Real-time optimal power flow using twin delayed deep deterministic policy gradient algorithm. IEEE Access **8**, 213611–213618 (2020). doi:10.1109/ACCESS.2020.3041007

[23] Nie, H., Chen, Y., Song, Y., Huang, S.: A general real-time opf algorithm using ddpg with multiple simulation platforms. In: 2019 IEEE Innovative Smart Grid Technologies - Asia (ISGT Asia), pp. 3713–3718 (2019). doi:10.1109/ISGT-Asia.2019.8881174

[24] Zhou, Y., Lee, W.-J., Diao, R., Shi, D.: Deep reinforcement learning based real-time ac optimal power flow considering uncertainties. Journal of Modern Power Systems and Clean Energy, 1–11 (2021). doi:10.35833/MPCE.2020.000885

[25] Li, H., He, H.: Learning to operate distribution networks with safe deep reinforcement learning. IEEE Transactions on Smart Grid, 1–1 (2022). doi:10.1109/TSG.2022.3142961

[26] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc., (2019). http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[27] Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. arXiv (2014). doi:10.48550/ARXIV.1412.6980. https://arxiv.org/abs/1412.6980

Statement of Originality

I hereby confirm, that I have written the accompanying thesis by myself, without contributions from any sources other than those cited in the text and acknowledgments. I certify, that I have followed the general principles of scientific work and publications as written in the guidelines of good research of the Carl von Ossietzky University of Oldenburg. This work has not yet been submitted to any examination office in the same or similar form.

Oldenburg, May 25, 2022 — Fenno Boomgaarden