TECHNISCHE UNIVERSITÄT WIEN

Institut für Computertechnik
Institute of Computer Technology

A MASTER THESIS ON

# Simulative Optimization of Energy Communities based on Reinforcement Learning

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

Diplom-Ingenieur

(Equivalent to Master of Science)

in

Embedded Systems (066 504)

by

**Maren Konrad**

12011863

**Supervisor(s):**

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Thilo Sauter

Univ.Ass. Dipl.-Ing. Thomas Leopold, BSc

Projektass. Dipl.-Ing. Stefan Wilker, B.Eng.

Vienna, Austria

August 2023

# Abstract

Energy generation still produces $CO_2$ and pushes the climate crisis. By grouping energy consumers and prosumers into energy communities, these can share energy within the community and help to reduce emissions. Due to weather conditions, fluctuations in generation of renewable energy are natural. There are certain flexibilities of renewable energy source like reducing the photovoltaic plants production and controlling the consumption of heat pumps within a specific range. Along with energy storage systems these flexibilities can be used to optimize the energy community. There are various objectives for an optimization, but in this work the main goals are to minimize the total grid load on the transformer and to smooth the load peaks. Within the scope of a project the optimization was implemented as a mixed integer linear approach. This work tackles the aim of using a deep reinforcement learning method to optimize energy communities. The method should learn to determine the flexibilities of photovoltaic plants, heat pumps and batteries to archive the main optimization goals. On top of a base implementation of the reinforcement learning optimization, various improvements are done. The reinforcement learning optimization contains two important components, environment and agent. The experiments are based on each other, and modifications are implemented in environment and agent of the optimization. The results of the experiments are showing an enhancement of the reinforcement learning optimization. To achieve a comparable optimization to the linear one, further modifications and a change of training data are necessary. However, this thesis presents a working concept for the use of deep reinforcement learning in the context of optimization in energy communities.

# Kurzfassung

Die Erzeugung von Energie produziert noch immer $CO_2$ Emissionen und treibt damit die Klimakrise weiter an. Verbraucher und Prosumer können in Energiegemeinschaften gruppiert werden. In den Energiegemeinschaften kann die produzierte Energie verteilt werden und damit zu einer Emissionsreduktion beitragen. Schwankungen in der Produktion von erneuerbaren Strom treten aufgrund von Wetterbedingungen natürlich auf. Es gibt gewisse Flexiblititäten bei erneuerbaren Energiequellen, wie die Verringerung der Produktion durch Photovoltaik Anlagen oder die Kontrolle des Verbrauchs von Wärmepumpen in einem vorgegebenen Bereich. Zusammen mit Energiespeichern können diese Flexibilitäten genutzt werden um die Energiegemeinschaften zu optimieren. Für die Optimierung gibt es verschiedene Ziele. In dieser Arbeit sind die Hauptziele die Minimierung der Netzleistung am Transformator und die Glättung von Lastspitzen. Im Rahmen eines Projektes wurde die Optimierung als gemischt-ganzzahliger linearer Ansatz umgesetzt. Das Ziel dieser Arbeit ist die Optimierung der Energiegemeinschaften mit einer Deep Reinforcement Learning Methode. Die Flexibilitäten von Photovoltaik Anlagen, Wärmepumpen und Batterien sollen von der Methode bestimmt werden um die Hauptziele der Optimierung zu erreichen. Ausgehend von einer Basisimplementation der Reinforcement Learning Optimierung werden verschiedene Verbesserungen vorgenommen. Die Reinforcement Learning Optimierung besteht aus zwei wichtigen Bestandteilen, der Environment und dem Agent. Die Experimente bauen aufeinander auf und Modifikationen sind in Environment und Agent implementiert. Die Ergebnisse der Experimente zeigen eine Verbesserung der Reinforcement Learning Optimization. Damit eine vergleichbare Optimierung zur linearen Optimierung erreicht werden kann, sind weitere Modifikationen und ein Wechsel der Trainingsdaten nötig. Jedoch stellt diese Arbeit ein funktionierendes Konzept für den Einsatz von Deep Reinforcement Learning im Kontext der Optimierung von Energiegemeinschaften vor.

# *Erklärung*

*Hiermit erkläre ich, dass die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt wurde. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.*

*Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.*

# Copyright Statement

I, Maren Konrad, hereby declare that this thesis is my own original work and, to the best of my knowledge and belief, it does not:

- Breach copyright or other intellectual property rights of a third party.
- Contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
- Contain material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning.
- Contain substantial portions of third party copyright material, including but not limited to charts, diagrams, graphs, photographs or maps, or in instances where it does, I have obtained permission to use such material and allow it to be made accessible worldwide via the Internet.

Signature:


Vienna, Austria, August 2023

# Acknowledgment

First, this work was only possible with Professor Thilo Sauter, the ICT and Energy&IT group. Thank you for the opportunity to do this work! A special thanks to Valentin Bauer and Thomas Leopold for creating the base for my work as the originator of the energy community controller and the mixed integer linear optimization, for helping me with my problems and answering my open questions. Thank you, Stefan Wilker, for the support and feedback whenever I needed it.

Through my ups and downs, my Viennese "family" has had my back, thank you for being there for me!

Thanks Dad! For being my role model, giving me guidance and supporting me!

# Contents

**Abstract** iii

**Kurzfassung** iv

**1 Introduction** 1

1.1 Research Questions . . . 2

1.2 Outline . . . 3

**2 State of the Art** 5

2.1 Deep Learning . . . 5

2.2 Reinforcement Learning . . . 6

2.3 Reinforcement Learning in Energy Sector . . . 10

2.4 Limitations and Scope . . . 12

**3 Methodology** 15

3.1 Overview . . . 15

3.2 Constraints and Bounds . . . 16

3.3 Reinforcement Learning Method . . . 18

3.4 Environment . . . 19

3.5 Reinforcement Learning Agent . . . 23

3.6 Interaction of method with the framework . . . 25

**4 Training and Results** 27

4.1 Experiments . . . 27

4.2 Result of optimization cases . . . 31

**5 Discussion** 45

5.1 Analysis of the Characteristics . . . 45

5.2 Experiments . . . . . 47
5.3 Discussion of RL optimization . . . . . 53

**6 Conclusion and Outlook** **57**
6.1 Conclusion . . . . . 57
6.2 Outlook . . . . . 58

**Bibliography** **60**

# List of Tables


3.1 Variable conversion from EC state to environment state . . . . . 21
3.2 Overview of the variables to be normalized with their value range and normalization range . . . . . 22
3.3 Overview of the actions with their value range as well as the normalization range . . . 22

4.1 Baseline parameters of the DDPG agent . . . . . 28
4.2 Overview of the optimization case and the corresponding unique identifier . . . . . 32
4.3 Overview and Explanation of the used KPIs . . . . . 36
4.4 KPIs for no optimization, mixed integer linear optimization and RL optimization cases: baseline, replace normalization and batch normalization . . . . . 37
4.5 KPIs for RL optimization cases: separate PV systems, dependent reward combinations and the verification . . . . . 38

5.1 Selected KPIs for RL optimization experiments . . . . . 48
5.2 Selected KPIs for RL optimization validation . . . . . 54

# List of Figures


2.1 Network topology for deep learning and the simplified structure of the human brain . 6
2.2 Based on Sutton and Barto [1], the interaction of agent and environment in a RL system 7
2.3 Selected overview of RL methods based on the book 'Deep Reinforcement Learning' [2] 9
2.4 Selected methods with regards to of the policy based on Dong et al. [2] . . . . . 10

3.1 Interaction of the Bifrost *core* with the EC optimization . . . . . 16
3.2 Structure of the actor network . . . . . 24
3.3 Structure of the critic network . . . . . 24
3.4 Sequence diagram of interaction between framework, environment and RL agent . . . 26

4.1 Overview of the dependence of the experiment variations . . . . . 27
4.2 Structure of the actor network after replacing the input state normalization . . . . . 29
4.3 Structure of the critic network after replacing the input state normalization . . . . . 29
4.4 Structure of the actor network with batch normalization . . . . . 29
4.5 Structure of the critic network with batch normalization . . . . . 30
4.6 Explanation of the violin plot based on [3] . . . . . 33
4.7 Transformer Load for NoOpt, LinOpt, Base, Norm, BatchN, SepPV and Re1a . . . . . 39
4.8 Transformer Load for Re1b, Re2b and RLOpt . . . . . 40
4.9 Power distribution on the transformer for no optimization, mixed integer linear optimization and RL optimization cases: baseline, replace normalization and batch normalization . . . . . 41
4.10 Power distribution on the transformer for RL optimization cases: separate PV systems, dependent reward combinations and verification . . . . . 41
4.11 Power difference distribution on the transformer for no optimization, mixed integer linear optimization and RL optimization cases: baseline, replace normalization and batch normalization . . . . . 42

4.12 Power difference distribution on the transformer for RL optimization cases: separate PV systems, dependent reward combinations and verification . . . . . 42
4.13 Reward Functions of RL optimization . . . . . 43

5.1 Explanation of the distribution of transformer power . . . . . 46
5.2 Power distribution on the transformer for discussed experiments . . . . . 49
5.3 Transformer Power of Base and Norm prepared for discussion . . . . . 50
5.4 Transformer Power of Norm and BatchN prepared for discussion . . . . . 50
5.5 Transformer Power of BatchN and SepPV prepared for discussion . . . . . 51
5.6 Transformer Power of SepPV and Re1b prepared for discussion . . . . . 51
5.7 Transformer Power of Re1b and Re2b prepared for discussion . . . . . 52
5.8 Reward functions, worst and best case . . . . . 52
5.9 Reward functions with equal level . . . . . 53
5.10 Reward functions with light learning process . . . . . 53
5.11 Transformer Power of RLOpt and LinOpt prepared for discussion . . . . . 54
5.12 Transformer Power of NoOpt and LinOpt prepared for discussion . . . . . 54
5.13 Power distribution on the transformer of RLOpt and Re2b . . . . . 55
5.14 Transformer Power of RLOpt and Re2b prepared for discussion . . . . . 55

# Acronyms

**A3C** asynchronous advantage actor-critic. 9, 12, 18

**API** application programming interface. 14, 59

**Base** RL Optimization - Baseline. xiii, xiv, 32, 33, 39, 43, 47, 48, 49, 50, 52, 53, 57

**BatchN** RL Optimization - Batch Normalization. xiii, xiv, 32, 34, 39, 43, 47, 48, 49, 50, 51, 52

**DDPG** deep deterministic policy gradient. xi, 9, 12, 18, 23, 24, 25, 28, 29, 59

**DDQN** double deep Q-network. 11, 18

**DQN** deep Q-network. 9, 11, 12, 18, 24

**EC** energy community. xi, xiii, 2, 13, 14, 15, 16, 19, 20, 21, 22, 23, 25, 30, 32, 33, 34, 36, 57, 58, 59

**EV** electrical vehicle. 10, 11, 18

**F** flexibility. 17

**HAD** highest allowed demand. 17, 20, 21, 22

**IQR** interquartile range. 32, 46

**KPI** key performance indicator. xi, 31, 32, 33, 34, 35, 36, 37, 38, 45, 46, 47, 48, 49, 52, 54, 55

**LAD** lowest allowed demand. 17, 20, 21, 22

**LinOpt** Mixed Integer Linear Optimization. xiii, xiv, 32, 33, 39, 53, 54

**MCL** maximum calculated load. 21, 22

**MDP** markov decision process. 13

**MED** maximum energy demand. 17, 22

**MES** maximum energy supply. 18, 22

**NoOpt** No Optimization. xiii, xiv, 32, 33, 39, 53, 54

**Norm** RL Optimization - Replace Normalization. xiii, xiv, 32, 34, 39, 43, 47, 49, 50, 52, 53

**PE** predicted energy. 17

**PPO** proximal policy optimization. 8

**PV** photovoltaic. 1, 2, 14, 16, 17, 19, 20, 21, 22, 24, 28, 29, 30, 31, 34, 35, 36, 45, 46, 47, 48, 49, 50, 51, 53, 58

**Re1a** RL Optimization - Dependent Reward 1a. xiii, 32, 34, 39, 43, 47, 48, 49, 51, 52, 53

**Re1b** RL Optimization - Dependent Reward 1b. xiii, xiv, 32, 35, 40, 43, 47, 48, 49, 51, 52, 53

**Re2b** RL Optimization - Dependent Reward 2b. xiii, xiv, 32, 35, 40, 43, 47, 48, 49, 51, 52, 53, 55

**ReLU** rectified linear unit. 23

**RL** reinforcement learning. x, xi, xiii, xiv, 2, 3, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 18, 19, 22, 23, 25, 26, 27, 28, 30, 31, 32, 33, 34, 37, 38, 41, 42, 43, 47, 48, 52, 53, 54, 55, 57, 58, 59

**RLOpt** RL Optimization - Verification and Validation. xiii, xiv, 32, 35, 40, 43, 53, 54, 55, 58

**SE** stored energy. 18, 20, 21

**SepPV** RL Optimization - Separate PV Systems. xiii, xiv, 32, 34, 35, 39, 43, 47, 48, 49, 51, 52, 53

**SoC** state of charge. 20, 21, 22

**tanh** hyperbolic tangent. 24

**ToC** total capacity. 18, 20, 22

**TRPO** trust region policy optimization. 8

# Chapter 1

# Introduction

In current times sustainable and affordable energy production is on everybody's mind. Two of the reasons for this are the climate crisis and the increasing electricity prices.

The climate crisis is a continuous process, which is now ongoing for years. Slowing down the process can only effectively be managed by reducing the $CO_2$ emissions [4].

Fossil fuels like coal, natural gas and oil still account for around 35% of the $CO_2$ emissions from electricity and heat production in the energy generation share all over the world. The highest emissions are generated by coal-fired power plants. The most obvious solution for the $CO_2$ emission reduction would be to change from coal based energy production to renewable energy sources, like photovoltaic (PV) plants, wind turbines and hydro generation. [5]

Although more and more renewable energy is generated, it does not cover the demand. After the COVID-19 pandemic, the industry slowly recovered in 2021 and this resulted in increased energy demand and rising electricity prices [6]. Due to the war in Ukraine the natural gas dependence on Russia and therefore the increasing prices for the gas resulted into another increase of the electricity costs in 2022 in Europe [7].

Integrating renewable energy sources into the electrical grid is not only beneficial for reduction of the $CO_2$ emissions, it also raises concerns. Uncertainty in weather, wind speed and solar irradiation affects the renewable energy generation. This could result in voltage and frequency fluctations, low inertia and fault ride through capability issues and makes it hard to maintain a flexible, stable and reliable grid with renewable energy resources. Advanced control methodologies and energy storage devices are two important countermeasures for these issues. [8]

The increasing electricity demand and the goal to reduce emissions will be grid expansion and integrating more renewable energy sources. As grid expansion is cost intensive and the problems with renewable energy sources are mentioned above, distributed local energy generation will spread.

Local energy communities (ECs) can help reducing the emissions and by sharing energy within the community, the costs for electricity can be decreased. Not only these two points are in focus, also the self-sufficiency and energy supply security are of interest for those communities. On the other hand, the distribution systems operators represent other interests like the avoidance of grid congestion and the deferral of network investments. This can be achieved by local balancing, load uniformity, activation of flexible generation and demand during the day to avoid load peaks. However, fluctuations in demand and generation are naturally due to renewable energy sources and weather conditions. To achieve the goals of local balancing and load uniformity the flexibility of renewable energy sources and energy storage systems can be used. In order to do this, an extensive optimization depending on the current load, the weather, the demand and generation is necessary. [9]

In Austria renewable ECs (Erneuerbare Energie Gemeinschaften) are regulated by law. The ECs are allowed to generate, store, consume and sell energy from renewable sources. Not only, the use of renewable energy is an advantage of these ECs, also financial incentive as reduction of grid fee. [10]

However, the use of the renewable energy within the ECs are not guaranteed as there is currently no optimization of the energy share, due to a lack of data. Participants do not share data about consumption and production and there is currently - as of the implementation of the law - no other way to get information about the availability of renewable energy within the EC.

In the context of this work ECs are a group of consumers and prosumers. The consumers contribute mostly passively, as they only consume energy, whereas the prosumers do both, consume energy and also produce energy with renewable energy sources. The settlement, used in the work, contains 20 households with different load profiles, five commercial buildings, two communal buildings and one community battery. Ten of the households own a heat pump and four houses have a PV installation. Besides the load profile the weather data is relevant, as the generation of PV energy and the demand of the heat pumps depends on the weather.

## 1.1 Research Questions

Various goals of ECs are described in the previous section, which can be optimized. For this work, the focus of the optimization is to minimize the total grid load and smoothing the peaks in the load of the EC. The current implementation uses a mixed integer linear approach for the optimization. The goal of this work is to investigate, if the realization can be done with a reinforcement learning (RL) method. Accordingly the main research question is: 'Can the grid load be minimized and the peaks in the load smoothed by optimizing an EC with a RL algorithm?'

Some additional questions are derived to enclose the scope of this thesis:

- Which RL variant might solve the optimization problem best?
- Can the minimized load and the smoothed load peaks of the RL optimization be compared with an already implemented mixed integer linear optimization?
- Are there further improvements for the RL method in order to get closer to the optimization goal of minimizing the grid load and smoothing the peaks in the load?

## 1.2 Outline

In this master thesis, chapter 2 guides into the field of deep learning and RL in a common sense and specific into the energy sector, but also covers the given limitations within this work. Chapter 3 describes the implementation of the RL optimization and its integration into the used framework. Afterwards the results of different optimization cases are presented in chapter 4. The presented results are compared and discussed in chapter 5. The thesis ends with a conclusion and an outlook for further works in chapter 6.

# Chapter 2

# State of the Art

This chapter gives an overview of deep learning, RL and the combination of both, deep RL. Besides a general view on these topics, also a more specific overview on the usage of (deep) RL in the energy sector is to be imparted. Afterwards the restrictions and limitations within this thesis shall be addressed and the used framework is introduced.

## 2.1 Deep Learning

Rosenblatt makes the analogy between the network topology and the human brain already in 1961 [11]. Based on this analogy the following comparison is made. The brain cells are interconnected by synapses which enable the exchange of information. During a lifespan these connections can change depending on the experience of a person. As an example, the image of a flower can be captured by a persons eye. Through nerves and synapses this image is transmitted into the brain. Depending on the knowledge of the person, it can be categorized as either a specific kind of flower, a flower or no flower at all. This setup is simplified illustrated on the right side of fig. 2.1. Transferring this example into deep learning, the network topology for this sample is depicted on the left side of fig. 2.1. From the eye to the brain, the image of the flower goes through multiple cells and synapses. Each of the passed cells will be called a layer in deep learning. There are always at least three layers in deep learning which are summarized as network. The input layer, the output layer and one or more so-called hidden layers. Starting with the input layer, the comparison can be the image captured by the eye and the kind of flower recognized by the brain will be the output layer. As the image passes cells between the input and the output, all these cells in between will be called hidden layers. A further relation between the human brain and deep learning will be the synapses and the weights. Due to experiences a person could learn the kind of flower from a previous unknown flower by adaptation of the synapses. Same could be noticed in the learning process of a network. The weights, which are connecting the layers, can be updated, which

could result in changed handling of the inputs and therefore another output results [12].

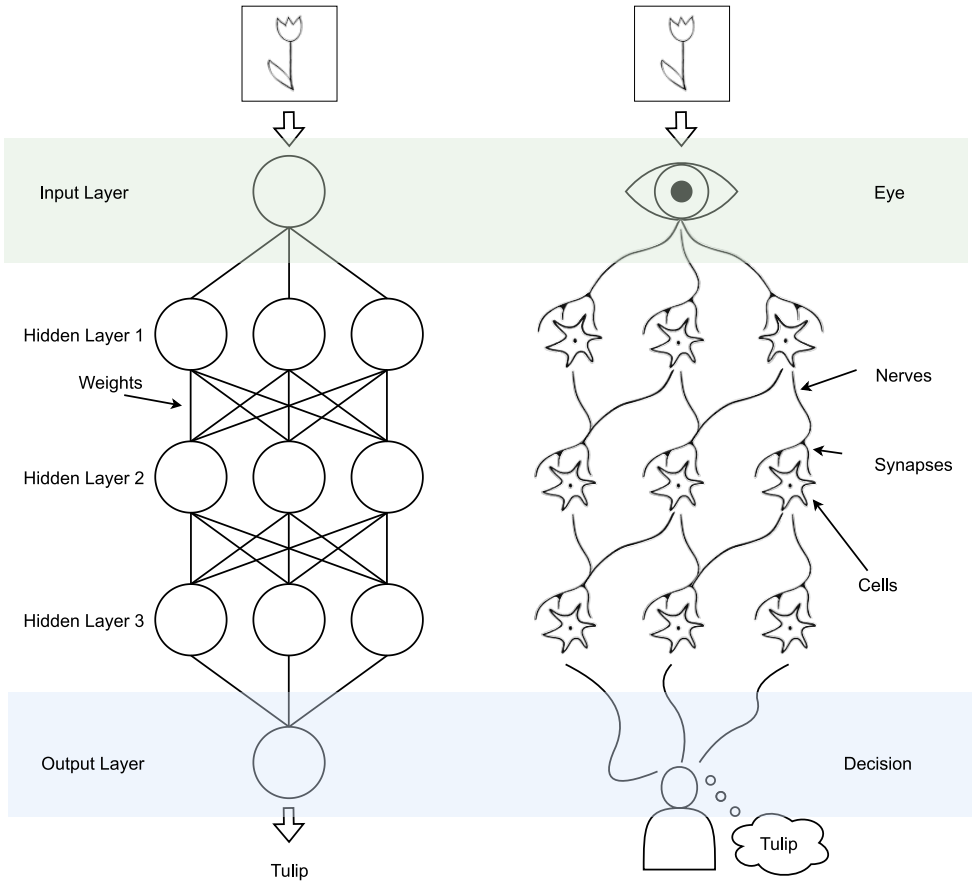



Figure 2.1: Network topology for deep learning and the simplified structure of the human brain

Besides the layers, a neural network also contains a loss function. This function is a metric to measure how well the network approximates the output regarding the input. There are several functions which can be used for the loss function, e.g. the mean squared error loss, the cross entropy or the mean absolute error. [13]

The weights can be updated not only through the computations flow forward from input to output, but also backward to optimize the loss function. For this purpose, the calculated gradients of the error derivatives are back-propagated towards the input layer [12].

## 2.2 Reinforcement Learning

Not only deep learning can be compared with human environment, also RL can be. The interaction with other people is shaped by the reaction and the behaviours of the counterpart. One is building up the knowledge of conversations with learning by doing. Depending on the reaction of other persons one can gauge if the participation in the talk is received well, and so the knowledge is updated with each conversation. There are computational approaches to learn from interaction like the field of machine

learning. This field is mainly divided into three sections. The focus of this thesis is RL, but before going into more detail about this, a short overview about supervised and unsupervised learning will be given, to distinguish between them. [1]

For supervised learning not only the input is provided in the training set but also the expected output is given. For example a training set of images is only given with labels which are expected on the output. This means that the system learns to label other but similar input correctly. [1] Using the example of the flower, in the training set the image of the flower would be labeled with the flower's name.

Without knowing the output labeling, the principle is to find and learn hidden structure in the given training set and apply it later on other data. This is called unsupervised learning. [1] Going back to the example, the image would be unlabeled, and it could be a training set full of images with different flowers, meaning that the algorithm learns to recognize a flower.

In contrast to supervised and unsupervised learning, RL does not need training data, instead the essential features are trial-and-error search and rewards. The two defining parts of a RL system are the agent and the environment, the latter represents the learning surrounding, e.g. a game or a simulation. Through the learning process the agent interacts with the environment to maximize the reward. The reward is given from the environment on performing an action at the state (see figure 2.2) and depends on how good the action in the situation was. Nevertheless this procedure is challenging as it is a fine line between exploration and exploitation. To tackle the goal of maximizing the reward the RL agent should use actions from before, where the previous reward was high. In order to find such actions, the agent needs to try actions it has not tried before. [1]

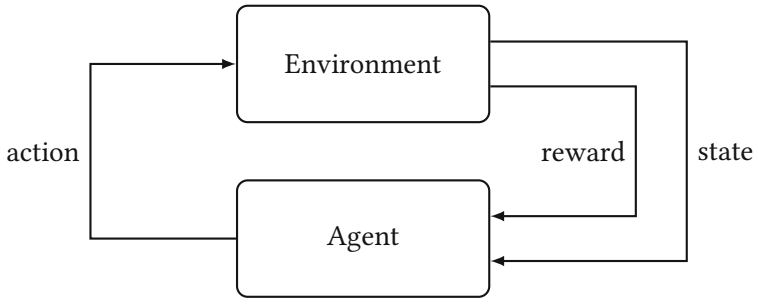



Figure 2.2: Based on Sutton and Barto [1], the interaction of agent and environment in a RL system

Due to memory and computational complexity, there are some limitations in RL [14]. On the one hand, this leads to limitations in the dimensions of states and actions, since tables are often used to map from a given state to an action [14]. Beyond the inherently low-dimensional problem of RL, RL is not sufficiently scalable [2]. In order to achieve a usable state of the environment for the agent, feature engineering must be performed by a developer, resulting in a lack of scalability and therefore

a limitation in complexity [2]. Development in deep learning provide new tools to overcome these restrictions [14]. The combination of using deep learning along with RL defines the field of deep RL [2]. One of the first successful union of both methods is summarized in a paper of 2013 by using raw pixels of Atari games for the state of the environment [15].

### 2.2.1 Components of Reinforcement Learning

In addition to the agent and the environment, RL also consists of four other elements, which are more deeply explained in Sutton and Barto [1]. In the following a short overview of the policy, the reward signal, the value function and the optional model of the environment are given and is based on Sutton and Barto [1].

For each time step the agent not only receives the state from the environment but also a reward. Over the long run, the agent wants to maximize the total reward, as the goal of the RL problem is defined by the reward signal. Besides that, the reward signal is the primary basis for altering the policy. Through a policy the states are mapped to actions in the states, so one could say the acting of the agent at a given time is defined by a policy. This indicates that the policy is the core of RL and it could be represented in various types as a simple function, a lookup table or a search process. A policy that establishes the immediate path of RL is important, but secondarily so is the long-term path. The advantage over the long run is specified by the value function. In that function the state which is most likely to follow and the corresponding rewards are taken into account. Possibly the current state can have low reward but a high value, if the following states have high rewards. This could be a faster way to achieve the RL goal. Values could be defined as predictions of rewards, which is the reason why it is harder to determine values than rewards, but makes the choice of the method to estimate values one of the most important components in RL algorithms. The last element and an optional one is the model of the environment. It is used for planning as it allows conclusions to be drawn about the behaviour of the environment. If a RL problem is solved by using a model it is called model-based method. The opposite to that would be model-free methods, which are simpler and trial-and-error based. [1]

### 2.2.2 Taxonomy of Reinforcement Learning

Depending on the perspective, RL can be categorized into various kinds [2]. Subsection 2.2.1 introduced model-based and model-free methods. Additionally, value-based methods and policy-based methods, as well as on-policy and off-policy methods, are discussed. A selected overview of the correlation between the different methods can be seen in fig. 2.3 [2]. The boxes present the various categories and the ellipses the concrete method. Proximal policy optimization (PPO) and trust region policy optimization (TRPO) as a direct predecessor of PPO, both are gradient-based methods. To keep the scope of the work, only a

limited amount of methods can introduced in more detail. Therefore the focus is on Q-Learning, deep Q-network (DQN), deep deterministic policy gradient (DDPG) and asynchronous advantage actor-critic (A3C) as they are already used in the energy sector. These methods are explained in section 2.3.

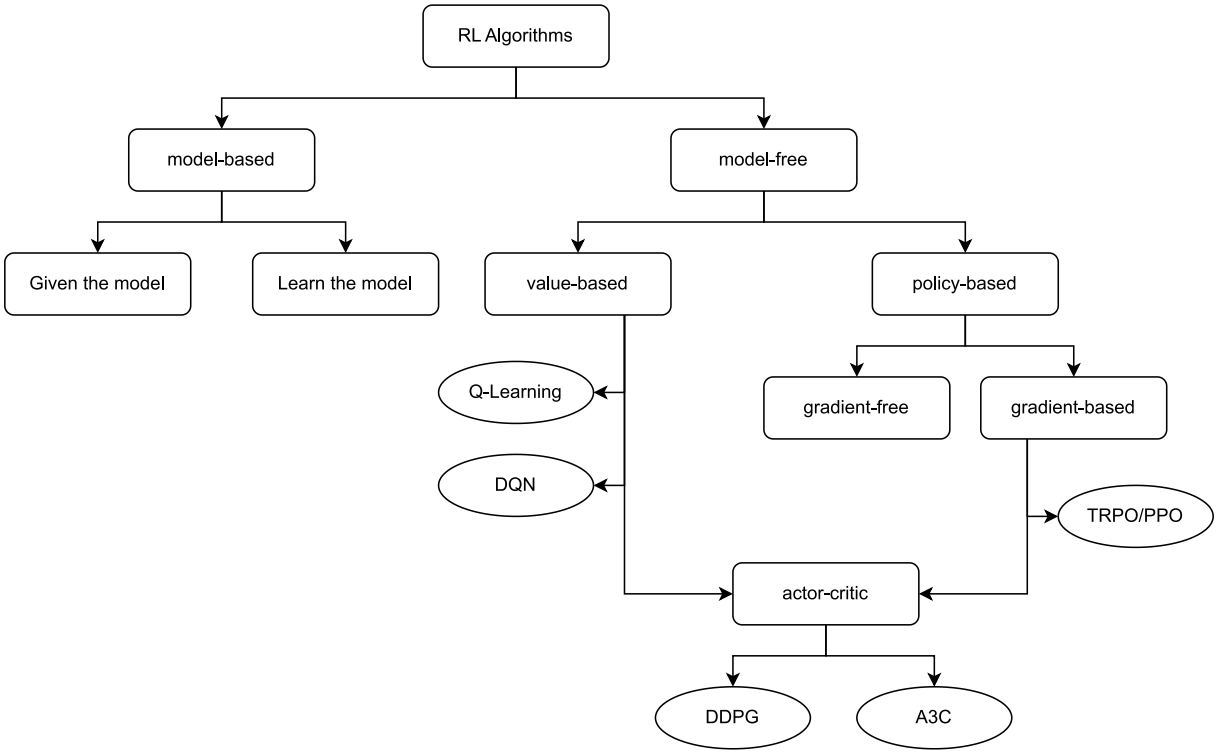



Figure 2.3: Selected overview of RL methods based on the book 'Deep Reinforcement Learning' [2]

Value-based methods use a value function to calculate the expected return when starting in a state, and the optimal policy has a corresponding function [14]. Through the corresponding function the policy is implicit as well as dependent from the value function and picks the action with the maximum value. The advantages of these methods are a small variance of the value function estimation, high sample efficiency and rarity to get trapped in local optima. However there are also some disadvantages like it can easily result in overestimation and usually cannot handle continuous action space. [2]

The policy-based methods do not maintain the value function and optimize the policy directly. Using either gradient-based or gradient-free optimization the parameters of the policy are updated to maximize the expected return [14]. Gradient-based optimization is mostly the method of choice for deep RL algorithms, because it is suitable for continuous or high-dimensional action space [2, 14]. The most popular method is created by combining the merits of the value-based and the policy-based method and is called actor-critic method [2]. To improve the sample efficiency the value-based method is used and the policy-based method supports with learning the policy function to be suitable for discrete or continuous action space [2]. Extending the standard policy gradients for stochastic policies to deterministic policies has the advantage that deterministic policy gradients, other than stochastic

policy gradients, only integrate over the state space, which requires fewer samples in problems with larger action space [14].

With regards to the policy two further subcategories are possible, on-policy and off-policy (see fig. 2.4). On-policy means that the policy interacting with the environment and the policy to improve is the same policy. This requires the agent to engage with the environment. Off-policy is improving a different policy than the policy that is used to generate the data. Which means that also different agents can interact with the environment and the results can all be used to improve the policy. [2]

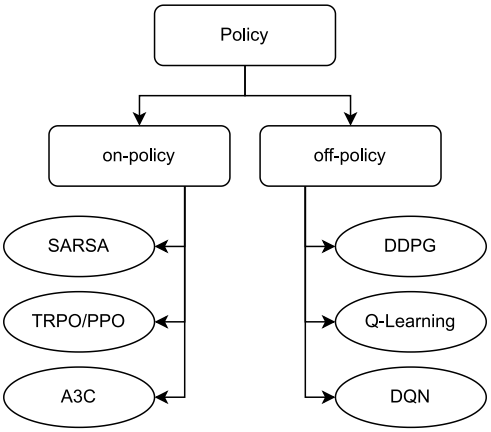



Figure 2.4: Selected methods with regards to of the policy based on Dong et al. [2]

## 2.3 Reinforcement Learning in Energy Sector

Following the introduction of RL, a state-of-the-art literature review in the energy sector is given.

### 2.3.1 Q-Learning

There are a few papers using the Q-learning algorithm. This methods aims to approximate the optimal Q-value function by storing all state-action value pairs in a table, which is updated at each time step [1].

In 'A Q-learning Method for Scheduling Shared EVs under Uncertain User Demand and Wind Power Supply' [16], the authors introduce shared electrical vehicles (EVs) as perfect users for wind power, however it is highly challenging because of the randomness in wind power supply and the user demand to charge the vehicle. The approach is to have an operation center in the building, which collects the local information about the wind power and the user demand. Each EV updates the state and chooses an action separately. The algorithm implemented in the paper increases the rate of fulfilling the user demand and keeps the usage of the wind power to a high level. [16]

The papers 'A Q-Learning Based Charging Scheduling Scheme for Electric Vehicles' [17] and 'Real Time Controlling Algorithm for Vehicle to Grid System under Price Uncertainties' [18] are focused

on Q-learning as well. In [17], the goal is to satisfy both EV users' and grid operators' needs and convenient usage. The focus lays on the bidirectional interaction between vehicle and grid and use time-of-use pricing, vehicle to grid capability and flexibility of charging as inputs [17]. In [18] the algorithm enables the frequency regulation services and controls the EVs charging and discharging according to the grid situation. The action space in all three papers is discrete and also the state space is manageable, due to Q-Learning.

### 2.3.2 Deep Q-Learning

The combination of deep learning and Q-learning called DQN is used in 'Multi agent deep Q-reinforcement learning autonomous low voltage grid control' [19]. In this paper, the authors simulate a low voltage grid environment with pseudo-measured data of household loads, EV charging and PV generation profiles. The goal of the algorithm is to reduce the maximum capacity of the transformer and though this also reduces the bidirectional power flow to the higher grid level. The results are promising and the next steps would be to use double and duel Q-learning algorithms. [19]

In DQN algorithm, the states are taken as input to the neural network, the network calculates the action values and chooses the maximum value as the output. This algorithm remains in a discrete action space. DQN may lead to overestimation during the learning process, which can be challenged by using a double deep Q-network (DDQN) with dueling network architecture. DDQN uses two neural networks, one is the target network and the other the main network. The main network receives the action with the highest value for the next state and also the Q-value of this action in the target network. In dueling DQN the output not only depends on the value function, but also on a advantage function. Where the value function relies on the current state, the advantage function also takes into account a certain action relative to other actions. [20]

DQN and DDQN are used in the paper 'Integration of Electric Vehicles in Smart Grid using Deep Reinforcement Learning' [21]. The algorithm's goal is maximizing the profit for the EV owner, by controlling the charge and discharge operation depending on the electricity price. The results of the DDQN method outperforms other state-of-the-art deep vehicle to grid systems. The randomness of new energy power generation systems make it hard to predict actions a system could perform. [21]

Therefore in 'Dueling Double Q-learning based Real-time Energy Dispatch in Grid-connected Microgrids' microgrid energy storage scheduling with wind power generation is discussed. The authors use a DDQN method with dueling network structure, but without a coordination between multiple agents. [20]

'Battery Control in a Smart Energy Network using Double Dueling Deep Q-Networks' as well covers the management of a storage system in a smart network. The proposed algorithm outperforms a model-

based benchmark, and can be further improved using exogenous data and forecasting. [22]

The three developed algorithms share the use of a discrete action space.

### 2.3.3 Asynchronous Advantage Actor-Critic

Switching to a continuous action space is only possible by changing the category, as introduced in subsection 2.2.2 so that an actor-critic method can be used. In [23] an A3C algorithm is used to solve the problem of the anti-peak characteristics of wind power with good convergence, stability and timeliness. A3C is based on the actor-critic method, however improves on it in three aspects. The network structure is optimized, by putting actor and critic network together. Then a asynchronous training framework is build up, which has a global network and a number of worker threads. Each thread is running independently and updates the global network after a time. The last aspect is to optimize the critic evaluation points. [23]

### 2.3.4 Deep Deterministic Policy Gradient

Another state-of-the-art method for continuous action spaces is DDPG. This algorithm again uses an actor and a critic network, as well as a replay buffer and Q-target approach which is derived from DQN [24].

There are various papers discussing DDPG in the energy sector. In one paper from Khooban and Gheisarnejad the DDPG is used to regulate the control parameters to stabilize the frequency in microgrids and was evaluated to be adaptive enough to fulfill the requirements [24].

In the scope of hydrogen-based systems, the optimal operation of these systems often neglect building thermal dynamics. Taking that into account while investigating the optimal operation problem is part of the paper 'Joint Optimization and Learning Approach for Smart Operation of Hydrogen-Based Building Energy Systems' and the authors also uses DDPG, but as an multi agent approach. Meaning that not only one agent is running to train the networks, but multiple ones. [25]

A multi agent DDPG method is also used by the authors of 'Multi-agent deep reinforcement learning-based approach for optimization in microgrid clusters with renewable energy'. The goal is full usage of the renewable energy, beside that the algorithm is also able to cope with generation and load uncertainty. [26]

## 2.4 Limitations and Scope

In this work are certain limitations given due to the broad field of RL. The scope of this work is briefly described in this section. The restrictions regarding the framework and simulation platform are de-

scribed.

### 2.4.1 Limitations in reinforcement learning

Most of the presented papers in section 2.3 and likewise the book of Sutton and Barto [1] and Z. Ding et al. [2] are describing the RL problem additionally as markov decision process (MDP). MDPs model the process of sequential decision making and are a mathematical abstraction of the problem. The naming for agent and environment as well as fig. 2.2 stem from the MDPs as a conceptual framework to define the problem of learning through interaction in order to fulfill a goal [1]. However, it is important to note that MDPs do not have significant relevance for this thesis as the RL problem is primarily addressed within the context of simulation, as the framework to work with is predefined.

Additionally, considering the amount of different (deep) RL methods available, it would exceed the scope of the thesis to address each algorithm comprehensively and discuss the presented methods in detail. In section 2.3 an broad overview of various methods used in the relevant literature is provided and the individual advantages and disadvantages of the methods are highlighted. This approach aims to highlight the rationale behind the choice of the specific method used in the implementation.

### 2.4.2 Used Framework

Testing grid infrastructure in the real world is often difficult, because it could affect critical infrastructure, has a direct impact on the residents, and due to the privacy laws in Europe data availability and resolution are strictly limited. To provide the possibility to test and demonstrate complete smart cities with a full integration of components, a framework to model and simulate grid infrastructure called Bifrost was developed. [27]

Bifrost's *core* functions as the central component are responsible for managing the simulation loop and maintaining the current state of the settlement. Other modules, like building model and weather models, can connect to the *core* and influence the simulation result by adjusting the dynamic state. The simulation loop of the *core* calls all registered modules in a deterministic order. [28]

In chapter 1, the problem of fluctuations in demand and generation due to renewable energy sources and weather conditions has already been discussed as well as some suggested solutions, like ECs and the optimization of these to balance the load.

As part of the research project cFlex (Community Flexibility in Regional Local Energy Systems) a mixed integer linear optimization within the EC controller has been developed. The cFlex project is funded by the Austrian Climate and Energy Fund (KLIEN) and carried out as part of the research program "Energieforschung (e!MISSION) 2018" (FFG #87657).

This Bifrost EC controller module can be used for simulating ECs and as a testing mechanism to minimize and smooth grid load. The controller predicts the consumption and production of the participants of the community, hands the prediction over to the community batteries and buildings, gets back the possible flexibility for each and adapts these, depending on some given restrictions from a higher level controller. [28]

Since the publication of the paper regarding the EC controller the development of the module is ongoing. Currently, the predicted values are passed on to a mixed integer linear programming method to enhance grid load optimization. The simulated ECs for this optimization includes uncontrollable loads (e.g. household consumption), batteries, heat pumps and PVs. Thereby the flexibility is composed of the range of the batteries, controlling of the heat pump consumption and cutting of the PV production.

The scope of this thesis is to compare the mixed integer linear approach for the optimization in the EC controller of the cFlex project with a RL approach. As a result, the simulation tool is limited to Bifrost, and an application programming interface (API) is provided for interaction with the EC controller. This thesis aims to compare and discuss the results obtained without optimization, with the mixed integer linear optimization method, and with the RL method.

# Chapter 3

# Methodology

This chapter gives an overview about the problem to solve and a detailed view into the implemented solution. This approach is further divided into the choice of the used RL method, the structure of and interaction between the environment and agent.

## 3.1 Overview

The proposed concept within this work is accompanied and carried out in the scope of the research project *AI-flex: Autonomous AI for cellular energy systems increasing flexibilities provided by sector coupling and distributed storage*[1].

Due to the project setting, the framework to use is already given and introduced in section 2.4.2. In this section, Bifrost as simulation platform and the associated EC controller module are described. The interaction of the EC controller with the Bifrost *core* is depicted in fig. 3.1. Within the simulation loop, the *core* calls the EC controller. The control unit of the controller requests predictions and calls the optimization. Here, the control unit queries the prediction of energy consumption and production and passes these values over to the optimization process. Once the flexibilities for the variable resources are computed, the optimized schedule is transmitted back to the control unit. The dynamic data of the Bifrost state is adjusted accordingly and returned to the *core*. Subsequently the Bifrost building model is called from the *cores'* simulation loop and the components of the settlement like households, commercial buildings and batteries are adapting the state optimized by the EC controller in order to archive the optimization goal.

Upon finalization of the cFlex project, the optimization process for the EC within the controller re-

[1]The project AI-flex has received funding in the framework of the joint programming initiative ERA-Net Smart Energy Systems' focus initiative Digital Transformation for the Energy Transition, with support from the European Union's Horizon 2020 research and innovation program under grant agreement No 883973. For more Information visit https://www.hsbi.de/forschung/forschungsprojekte/aktuelle-projekte-fb-3/haubrock-ai-flex

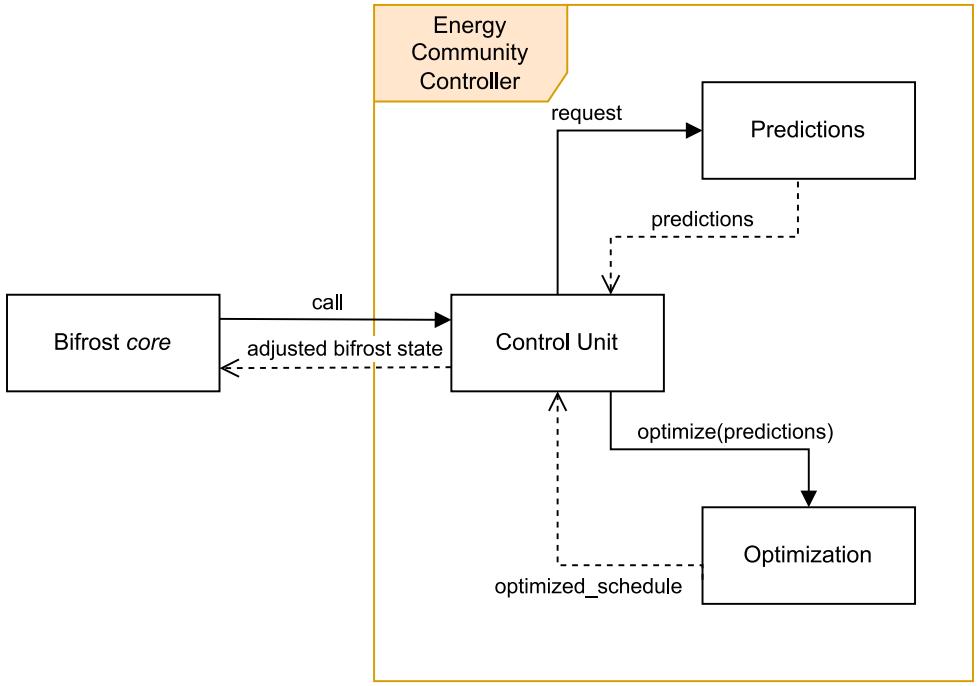



Figure 3.1: Interaction of the Bifrost *core* with the EC optimization

lied on a mixed integer linear programming approach. This methodology requires a substantial amount of code to represent the constraints and bounds for each variable. Additionally, these restrictions must be formulated as mixed integer linear problems, thereby requiring further effort. Reducing the manual complexity could be done with RL, whereas deep RL might be better, as it is possible to reduce manual feature extraction with deep learning [2] (see section 2.2). This leads to the main research questions, to investigate if the optimization of ECs, with the goal to minimize the grid load and to smooth load peaks, can be fulfilled with a RL approach (see section 1.1). For realization, the mixed integer linear optimization should be replaced by the RL optimization and the other surroundings in the EC controller stay the same.

## 3.2 Constraints and Bounds

In the mixed integer linear optimization, various types of energy resources, along with their predicted energy span as well as flexibilities, constraints and bounds are already predefined. In this section the relevant elements are introduced to provide a clearer understanding for the subsequent sections.

The four different types of energy appliances are PV systems, heat pumps, batteries and uncontrollable loads,. The latter summarize all remaining loads within the simulated EC. For each of these resources a predicted energy value exists for each time step. Positive values indicate energy production, while negative values indicate energy consumption. The flexibility operates in the exact opposite man-

ner. In this case, a positive value represents a reduction in production, while a negative value indicates a reduction of consumption.

For a better differentiation of the constraints and bounds a designation scheme is used. Therefore the first two letters specify the energy resource (*PV*: PV system, *HP*: heat pump, *BA*: battery), after a dash, either a *c* for constraint or a *b* for a bound with a unique number follows. For example the first constraint of the heat pump would be *HP-c1*.

**PV Systems** can only produce energy and never consume it. The flexibility (F) of this resource can only reduce the production as the produced energy mainly depends on weather conditions.

- **PV-c1** The production of the PV system can only be reduced to zero and not more than the predicted energy (PE) (see eq. (3.1)).
- **PV-b1** The PV system only can produce energy, so the flexibility value needs to remain positive (see eq. (3.2))

$$F \leq -PE \tag{3.1}$$

$$F \geq 0 \tag{3.2}$$

**Heat pumps** can only consume energy and never produce it. The consumption of the energy can be delayed or brought forward, which results in positive and negative flexibility for this resource.

- **HP-c1** The consumption cannot exceed the maximum energy demand (MED), but the predicted energy must be considered (see eq. (3.3)).
- **HP-c2** A heat pump can only consume energy, but can reduce the predicted energy demand to zero (see eq. (3.4)).
- **HP-c3** The highest allowed demand (HAD) restricts the flexibility range. So the consumption of the heat pump after applying the flexibility to the predicted value needs to be below this limit (see eq. (3.5)).
- **HP-c4** The lowest allowed demand (LAD) defines the lower limit of the flexibility range. The predicted energy can only be reduced by the flexibility till this limit is reached (see eq. (3.6)).

$$F \leq MED - PE \tag{3.3}$$

$$-F \leq PE \tag{3.4}$$

$$F \leq HAD - PE \tag{3.5}$$

$$F \geq LAD - PE \tag{3.6}$$

**Batteries** can consume and produce energy depending on the state of charge. However this results in a flexibility range from negative to positive values.

- **BA-c1** The battery can only be charged till the total capacity (ToC) is reached in consideration of the stored energy (SE) (see eq. (3.7)).
- **BA-c2** The battery can only supply energy until it is empty (see eq. (3.8)).
- **BA-b1** The consumption cannot exceed the maximum energy demand of the battery (see eq. (3.9))
- **BA-b2** The production cannot exceed the maximum energy supply (MES) (see eq. (3.8)).

$$F \leq ToC - SE \tag{3.7}$$

$$F \geq -SE \tag{3.8}$$

$$F \leq MED \tag{3.9}$$

$$F \geq MES \tag{3.10}$$

**Uncontrollable Loads** can consume energy. Every energy resource which does not belong to one of the above are allocated here. As this resource is uncontrollable they do not have a flexibility.

## 3.3 Reinforcement Learning Method

The wide field of RL makes it necessary to limit the considered number of methods to stay within the scope of this work. As a result of the literature review in section 2.3 the methods to evaluate have been reduced to the following: Q-learning, DQN, A3C and DDPG.

The results of the papers using Q-learning are promising. However, due to the Q-table the amount of states and actions are limited. To overcome this limitation, Q-learning can be combined with deep learning. This DQN method and its variants DDQN and dueling DQN is already used for smart grid applications like controlling batteries [21] or integrating EVs into the grid [22]. Both, Q-learning and DQN, are value-based methods and therefore limited to a discrete action space [2].

In this work, the state contains, among others elements, the predicted energy values, while the flexibilities are defining the actions that can be taken. This implies an continuous action space for the chosen method. Due to this, Q-learning and DQN are excluded from the potential methods. This leaves the actor-critic methods, A3C and DDPG, for consideration.

A3C uses an asynchronous training framework with different threads [23]. This results in a complexity that would exceed the scope of this work. After excluding all but one of the methods considered, only DDPG remains. This method is derived from DQN and consists of an actor and a critic network as well as a replay buffer [24]. The critic network learns using the same Q function as in DQN while

the actor network learns the policy function [2]. For updating the actor network a policy gradient algorithm is used, which applies the Q-value of the critic [2]. During the training process the balance of exploration and exploitation needs to be maintained, which is realized by adding noise to the actions [2]. For improving the stability of learning, a copy of the actor and critic network is used for calculating the target values [29].

## 3.4 Environment

An interface needs to be provided for the interaction between the RL agent and the simulation tool, because the predictions and relevant simulation parameters cannot be handled directly by the agent. This interface is offered by a customized environment within the RL optimization. The Bifrost state is handed over to the EC controller, which ensures the prediction of values for the optimization. TheEC state is based on the Bifrost state, but contains also predictions and is reduced to the relevant information for the optimization. The interface handles the conversion of the EC state and predictions regarding the environment state as well as the re-conversion with the optimized values. Additionally the evaluation of the last action is taking place in the environment, better known as the reward function. In the context of this chapter, the environment class is introduced. This includes an introduction of the chosen environment state and chosen actions and a deeper insight into the relevant functions.

### 3.4.1 State and Action Space

The environment state contains 14 values, which are explained in the following:

- Weather (3 values)
- Simulation time (4 values)
- PV energy (1 value)
- Battery (2 values)
- Heat pump (3 values)
- Uncontrollable load energy (1 value)

Each energy resource type is joined to one state representation to simplify the environment state. For example, eight heat pumps are represented as one in the environment state.

**Weather** For a better optimization of PV systems and heat pumps, the current temperature as well as the direct and diffuse solar irradiation is provided.

**Simulation time** The Bifrost simulation time is divided into day and weekday. To allow the RL system to use day-specific information, such as the weather and the season, the day is extracted out

of the time. Load-specific fluctuations are more likely to depend on the weekday. To convert day and weekday into cyclic data in the range [-1,1] for a suitable input for a neural network, trigonometric encoding is used [30]. Both, day and weekday, are then represented by two values, calculated of the original value using sin(x) and cos(x).

**PV energy** The relevant information about the PV system to extract out of the EC state and prediction is the predicted production of the system in the next time step.

**Battery** For the battery no predicted energy is available, as it can fully be used as flexibility. Thus, the values of the current time step are taken into account. Using the ToC and SE the state of charge (SoC) of the battery is calculated and also the measured energy is used in the state.

**Heat pump** The associated details regarding the heat pumps are once more the predicted energy. Besides that, also HAD and LAD are relevant as both give the range for the energy consumption in the next time step.

**Uncontrollable load energy** For the uncontrollable loads only the predicted consumption and production for the next time step is relevant. The predicted energy for all uncontrollable loads are summed up into one state value.

The actions are representing the flexibility values. The flexibility can only be modified for the PVs, batteries and heat pumps. Like for the state also for the action space each energy resource type is joined into one appearance. This leads to a representation of the action space by three continuous values with kWh as unit and their associated range:

- Battery flexibility with range [-1,1]
- PV flexibility with range [0,1]
- Heat pump flexibility with range [-1,1]

### 3.4.2 Conversion of States

This subsection explains the conversion between the EC and environment state in more detail and gives a brief insight into the normalization of the state.

The main difference between the two states is the representation of energy resources. In the environmental state, each resource type is represented once instead of individually representing each resource. This is achieved by combining the corresponding values from the EC state. Additionally, the day and weekday are derived from the timestamp provided by Bifrost. Table 3.1 lists all environment state variables along with their corresponding EC variables, as well as the units of the EC values.

The unit of values can influence the result of the neural network. To avoid this dependence normal-

| Environment State | EC State | EC Unit |
|---|---|---|
| Weather - temperature | temperature | °C |
| Weather - direct solar irradiation | direct solar irradiation | $W/m^2$ |
| Weather - diffuse solar irradiation | diffuse solar irradiation | $W/m^2$ |
| Time - day sin | Unix timestamp | seconds |
| Time - day cos | Unix timestamp | seconds |
| Time - weekday sin | Unix timestamp | seconds |
| Time - weekday cos | Unix timestamp | seconds |
| PV - energy | predicted energy | kWh |
| Battery - energy | measured energy | kWh |
| Battery - SoC | SE | kWh |
| Heat pump - energy | predicted energy | kWh |
| Heat pump - LAD | LAD | kWh |
| Heat pump - HAD | HAD | kWh |
| Load - energy | predicted energy | kWh |

Table 3.1: Variable conversion from EC state to environment state

ization can be used, which results in the acceleration of the learning phase of the network. Normalization transforms the data into a smaller range. In order to preserve the size ratio in the original data, the min-max normalization can be used. This linear transformation needs to know the current minimum and maximum value as well as the normalized range. The equation for calculating the normalized value is shown in eq. (3.11). [31]

$$value_{norm} = min_{norm} + \frac{max_{norm} - min_{norm}}{max_{value} - min_{value}} * (value - min_{value}) \tag{3.11}$$

In this work the range [-1,1] is used for the normalized data and the min-max normalization. For the de-normalization of the actions, eq. (3.11) is transposed to calculate the value in the original range.

The values in the environment state are all normalized with a few exceptions. No range is given for temperature, so a well-presented normalization is not possible. The maximum value for the solar irradiation is supplied by [32]. Also the time is not normalized, since sin(x) and cos(x) are already been applied by the conversion from the EC state and the range [-1,1] for the values is guaranteed. The upper bound for the uncontrollable energy load is unknown. To allow a scalability of the optimization for the normalization of the uncontrollable loads the maximum calculated load (MCL) is calculated. This calculations depends on the number of buildings and a theoretically maximum power demand per

building (assumed to be 25 kWh). The MCL is calculated for each time step by multiplying the number of buildings with the theoretically maximum power consumption. The range of the value as well as the normalization range for each variable are shown in table 3.2.

| Variable | Value Range | Normalization Range |
|---|---|---|
| Weather - direct solar irradiation | [0, 1361] | [0, 1] |
| Weather - diffuse solar irradiation | [0, 1361] | [0, 1] |
| PV - energy | [MES, 0] | [-1, 0] |
| Battery - energy | [MES, MED] | [-1, 1] |
| Battery - SoC | [0, ToC] | [0, 1] |
| Heat pump - energy | [0, MED] | [0, 1] |
| Heat pump - LAD | [-MED, MED] | [-1, 1] |
| Heat pump - HAD | [0, MED] | [0, 1] |
| Load - energy | [0, MCL] | [0, 1] |

Table 3.2: Overview of the variables to be normalized with their value range and normalization range

After normalizing the environment state, it is ready for further use in RL optimization. However, the results of the optimization have to be converted again, as they have to be fed back into Bifrost to apply them to the simulation. To do this, the flexibilities are converted from the normalization domain to the value range (see table 3.3). It should also be noted that for the environmental state, all energy resources of the same type are merged. So, in order to generate the EC state from the actions in the last step, they must be divided. Each individual resource receives a percentage of the total flexibility of all resources of the same type. For the PV systems, the percentage depends on the MES for each resource compared to the total MES over all PV plants. Similarly, for the batteries is the ToC used and for heat pumps the MED. Before the state is returned to Bifrost, the results are checked to see if they fall between the limits introduced in section 3.2.

| Action | Value Range | Normalization Range |
|---|---|---|
| Battery flexibility | [MES, MED] | [-1, 1] |
| PV flexibility | [0, -MES] | [0, 1] |
| Heat pump flexibility | [-MED, MED] | [-1, 1] |

Table 3.3: Overview of the actions with their value range as well as the normalization range

### 3.4.3 Reward Function

Another essential component in RL is the reward function. It is environment-specific, since the behavior of the optimization is defined here and cannot be located in the agent.

In this implementation, the reward calculation is divided into two parts, because of the given sequence of the simulation. In the first part, the actions are evaluated with respect to the constraints given in section 3.2. Therefore, the *flexibility reward* is introduced and set to zero at the beginning of the transformation from the actions to the EC state. A negative *flexibility reward* is added for each violation, and if not for a resource, the *flexibility reward* is increased. After the optimized EC state has been returned to the Bifrost *core*, the next simulation step is performed and then the optimization of the next time step is requested.

The new environment state and the previous one are used to judge the actions regarding to the optimization goals. One of them is to minimize the total grid load of the EC towards a setpoint, which is a certain power level the optimization tries to reach. This setpoint is assumed to be zero in this work, since the EC then means less work for the grid operator. The other objective is to smooth peaks in the grid load. For this purpose, the latter and the current environment state are compared and analyzed. The optimization goals are evaluated individually. If the restrictions are met, a positive reward is added, otherwise a negative one.

For the total reward returned to the agent, the *flexibility reward* and the reward from the optimization goals are summed up.

## 3.5 Reinforcement Learning Agent

The selected RL method (see section 3.3) is implemented as the agent and consists of three main components: the actor and critic network as well as the replay buffer. In this section the design of the separate components are explained and the interaction between them to build the DDPG agent.

### 3.5.1 Components

Layers and activation functions are used to build up the actor and critic network, but the network structure for both differ.

Due to the hidden layers in the network is it necessary to use an activation function to calculate the non-linear hidden layer values . In the literature it is recommended to use the rectified linear unit (ReLU) function, where no negative output values are possible (see eq. (3.12)). Depending on the output expectation of the network, an activation function is also applied here. [33]

$$g(z) = max\{0, z\} \tag{3.12}$$

**Actor network** - The input size for the actor network is defined by the size of the environment state and has an output for each continuous action. The selection of the activation function of the outputs depends on the expected actions. The flexibility of the PV system can only reach positive values, so a sigmoid function is chosen. The output values for that function are in the range [0,1]. Both, battery and heat pump flexibility, can reach positive and negative values, which makes a output range [-1,1] appropriate. Thus the decision is made to use the hyperbolic tangent (tanh) function. The exact structure of the actor network is depicted in fig. 3.2. Depending of the point of view, the numbers on the lines have two representations. The layer, where the arrow points towards, expects the given number as input variables. For example, the hidden layer expects 300 input variables. The layer, where the line starts, represents the other point of view. In this case, the number represents the number of neurons. To provide the 300 input variables for the hidden layer, the layer before needs also 300 neurons.

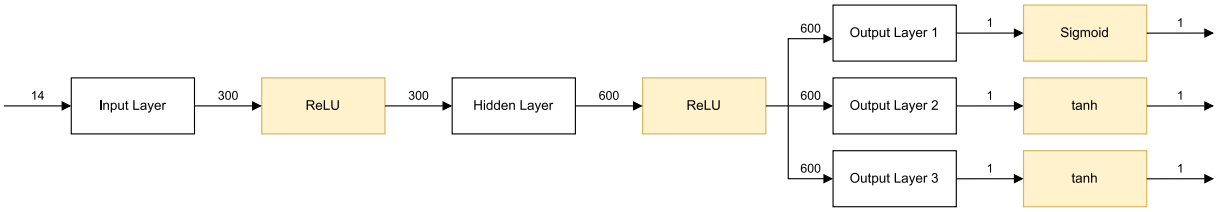



Figure 3.2: Structure of the actor network

**Critic network** - The whole critic network is shown in fig. 3.3. The aim of the network is to directly estimate the Q-function [2], for this reason it is not necessary to attach an activation function to the output layer. For predicting the Q-value, the environment state and the actions are necessary, this results in two input layers. To achieve one output value, both branches of the network are later added and further processed.

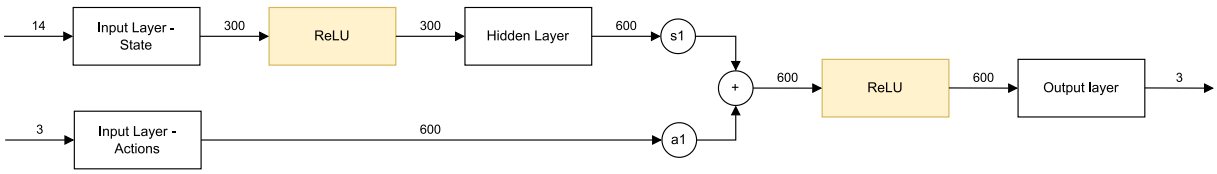



Figure 3.3: Structure of the critic network

**Replay buffer** - Samples are not independently distributed in the system, but most algorithms assume the opposite [29]. DQN addresses this issue by introducing a replay buffer, which is also used in DDPG [29]. Therefore, with each time step, a sample is saved in a predefined structure (state, action, reward, new state). If the buffer is full, the oldest sample is discarded [29].

**Noise** - For exploration in the training process the Ornstein-Uhlenbeck process [34] is used as suggested from Lillicrap et al [29].

### 3.5.2 Implementation

The implementation of the DDPG agent is derived from the corresponding paper [29]. At initialization, the replay buffer, an actor and a critic network are created, as well as a copy of them, called target network. To connect the environment with the agent, the latter must provide some functions.

In the training process the model is saved and can be loaded afterwards. Therefore saving and loading, each has a own function. Also the replay buffer needs to be updated at each time step, so there is a function to handle this update. The two main functions of the agent are *learn* and *choose_action*. Based on the current state, three actions determined by the actor network. In the training process some noise is calculated based on Ornstein-Uhlenbeck [34] and added to the original actions, in that way exploration is provided. For the learning process various steps are necessary. The target networks are used to get the Q-value and to calculate the target value. These networks are used to gain stability in the training process [29]. The target value is then used to calculate the loss with the mean squared error function. By reducing the loss the critic network is updated, while the update of the actor network is based on a policy gradient algorithm using the Q-value of the critic network. After performing this steps, both target networks are updated.

## 3.6 Interaction of method with the framework

The last step towards a fully working RL optimization for ECs is to link the environment and the agent.

The DDPG algorithm published in the associated paper [29] has the following execution order. First the state is received, then the action is selected and executed and the reward is observed. These information are then used for the learning process. In the given framework, the request for an optimization of the current time step is requested from Bifrost via the EC controller. This makes it necessary to slightly change the order for the algorithm. However, also the possibility to change the RL method should be enabled.

This splits the RL optimization into two main parts with two helper functions, which can be seen in fig. 3.4. The evaluation of the previous action is defining the first section, as this can only happen after applying the action to the simulation step. For that, the information about the current and last state as well as the previous action are used to calculate the reward and use it for the learning process of the RL agent. In the second part the actions for the current simulation step are chosen and applied to the simulation. The two helper function converting the EC state in the environment state and vice

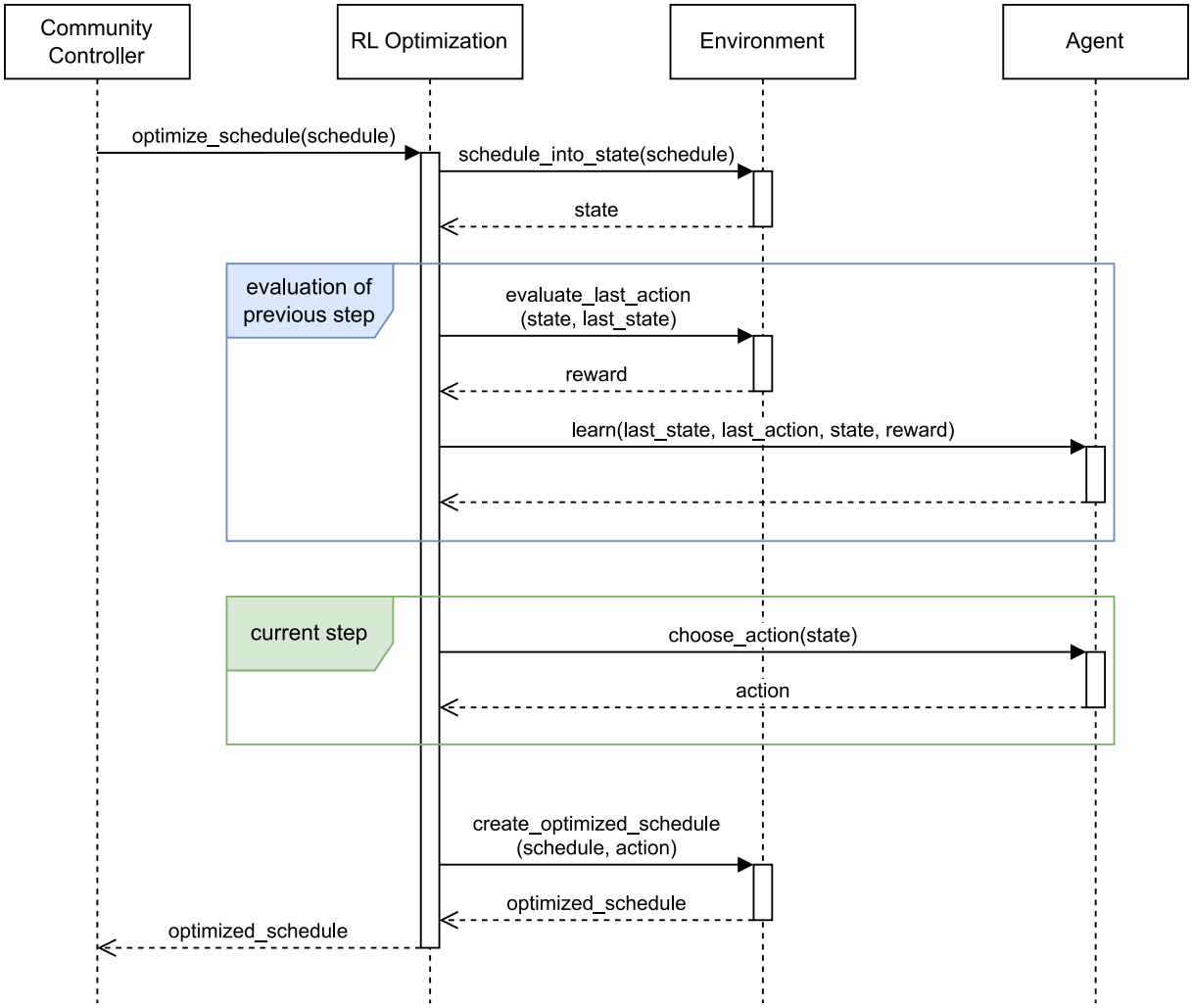



Figure 3.4: Sequence diagram of interaction between framework, environment and RL agent

versa.

By this, the RL method is well integrated into the Bifrost framework. The experiment details and results with the RL optimization are listed in the next chapter, followed by a comparison and discussion of the results.

# Chapter 4

# Training and Results

In this chapter the different variations within the training process of the RL optimization are introduced and also the results are presented. Not only the results of the RL optimization variants, but also the non optimized and linear optimized simulation results are presented to make a comparison of all optimizations possible.

## 4.1 Experiments

The baseline simulation is the implementation described in chapter 3. However, there are approaches to improve the baseline result by adjusting the agent or environment in the implementation, which are based on literature or additional considerations. The various experiments and corresponding arguments for simulation are introduced as part of this section, whereas the combination is shown in fig. 4.1.

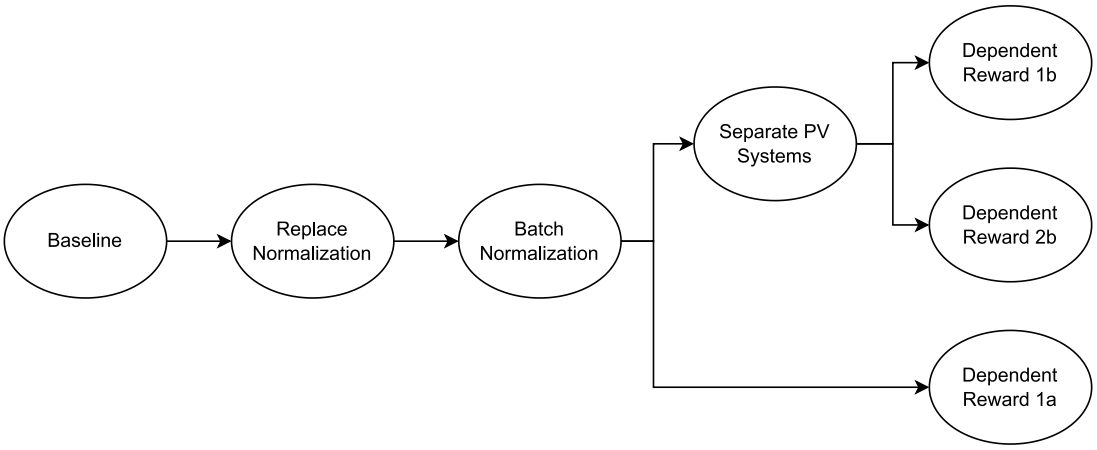



Figure 4.1: Overview of the dependence of the experiment variations

In order to not exceed the scope of the work, each experiment is simulated for training around 10.5 months and the first evaluation is based on that result. For the simulation are load profiles for

all buildings and weather data necessary. The limitation of 10.5 months is predefined by a lack of data availability. Due to the characteristics of RL this procedural is possible, as the model learns the behavior and tries to improve it with every time step. The best experiment results is then used for verification and validation. For this, the optimization is trained for one year, and evaluated with another year, where load profiles and weather data are different from the training year.

### 4.1.1 Baseline

The baseline of the RL optimization is the basic implementation of the environment and agent described in chapter 3. For a better comparison with the non optimized and the linear optimized case, the same settlement configuration is used. This settlement contains 20 households with different load profiles, depending on the residents, five commercial buildings, two communal buildings and one community battery. From all households, ten households own a heat pump and there are four PV installations. The parameters for the DDPG agent are shown in table 4.1.

| **Parameter** | **Value** |
|---|---|
| Batch size | 64 |
| $\gamma$ | 0.95 |
| $\tau$ | 0.001 |
| Learning rate - actor | 0.0001 |
| Learning rate - critic | 0.001 |
| Buffer size | 100000 |
| Gradient-based optimization | Adam [35] |

Table 4.1: Baseline parameters of the DDPG agent

### 4.1.2 Replace State Normalization

In section 3.4.2, the used min-max normalization for the input variables as well as their value range are explained. The min-max normalization is not ideal for the simulation, due to a few reasons. First, the temperature is not normalized, because there is no range given. On the other side, the used ranges for the normalization are not necessarily the maximal reached value for a variable, but the maximum possible value. This could result in inaccuracies. Another method to normalize the input states for the DDPG agent is batch normalization. This technique ensures effective learning across different types of units by normalizing the samples in a batch to unit mean and variance [29]. For both actor and critic network the min-max normalization is replaced by a batch normalization before the input layer (see

fig. 4.2 and fig. 4.3). The normalization of the input actions on the critic network is not necessary, as the action values are already in a normalized range. The numbers on the lines represent the expected input for the following layer, and the number of neurons the layer where the line starts.

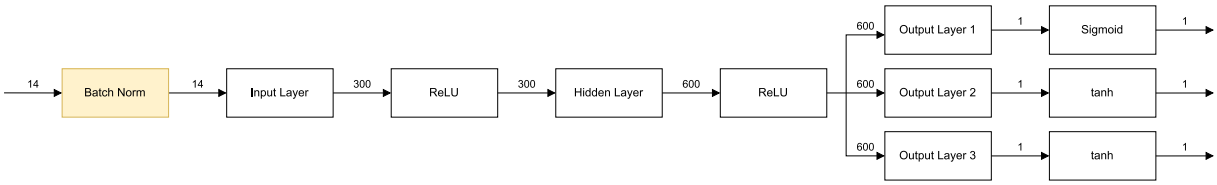



Figure 4.2: Structure of the actor network after replacing the input state normalization

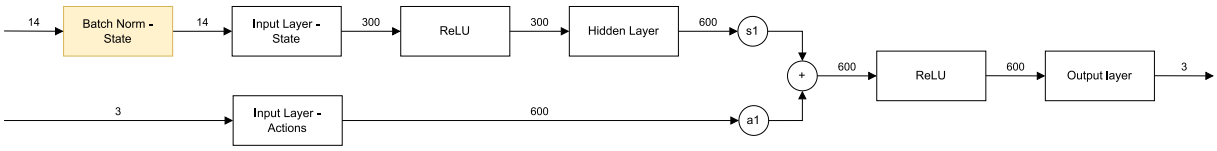



Figure 4.3: Structure of the critic network after replacing the input state normalization

### 4.1.3 Batch Normalization

In the paper regarding DDPG, the performance of this algorithm with and without batch normalization between the layers is compared. The result is that the use of batch normalization improves the performance. For this normalization the samples in a batch are normalized to unit mean and variance, but also guarantees balanced layer input to minimize covariance shift during training. [29]

Due to the outcome of the paper, one training variant is to add batch normalization to the neural network of the agent. The exact network layout of actor and critic are shown in fig. 4.4 and fig. 4.5.

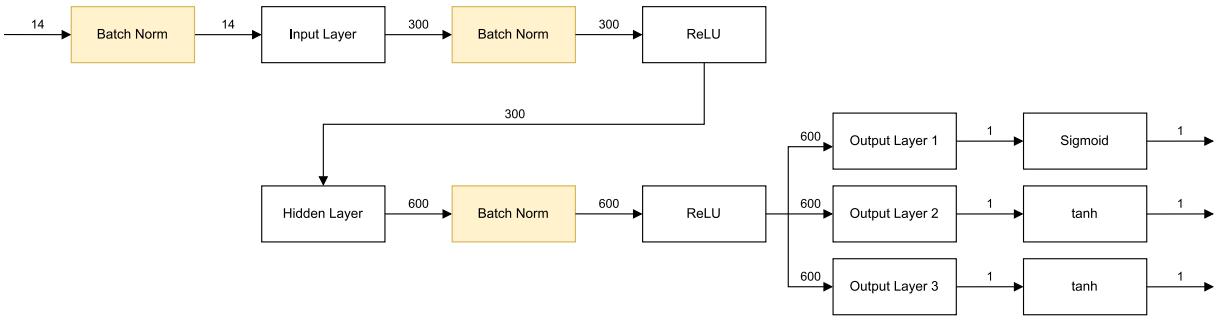



Figure 4.4: Structure of the actor network with batch normalization

### 4.1.4 Separate PV Systems

For the baseline, each resource type is added to one representing variable in the environment state. This works quite well for the battery and heat pump, but some accuracy is lost for the PV systems. The

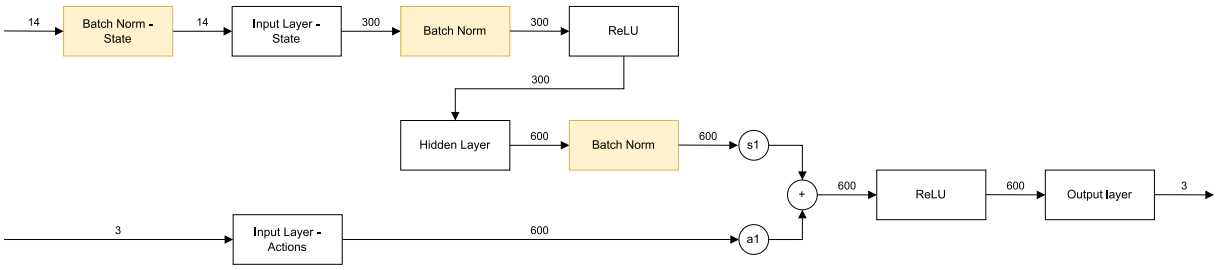



Figure 4.5: Structure of the critic network with batch normalization

energy production of a PV plant depends not only on the solar irradiation, but also on the orientation of the plant and the angle of irradiance. For this reason, one attempt is to adjust the environment state so that each PV system represents its own.

### 4.1.5 Dependent Reward Function

To achieve the optimization goal, the actions are judged by a reward function. One goal is to minimize the total grid load of the EC and another aim is to smooth peaks in the grid load. Meeting the restrictions adds a positive, constant number to the total reward, otherwise a negative, constant number is added. The implementation of the reward function is described in detail in section 3.4.3.

For this experiment the reward function is adjusted, whereas the *flexibility reward* is not changed. This makes it easier to determine if an adjustment improves the RL optimization. To make the reward dependent from the grid load and peak, the constant numbers are replaced by calculations and there are two different experiment variants, separated in *1* and *2*.

The deviation is calculated by subtracting the setpoint with the current grid load and getting the absolute value of it. The absolute value is used, as the difference from the setpoint is relevant and not the sign of the grid load. For smoothing peaks, the change is calculated by subtracting the previous grid load from the current one and again using the absolute value. Depending on the restriction, the reward is added to or subtracted from the total reward.

- **1** - In this experiment variant, the calculation of both rewards, deviation and change, are not longer fixed to a constant number. The calculation of the rewards is chosen in a way, that the reward for a value close to an expected good value has almost the same weight as before. This keeps the range of the reward more equal, but still judges outliers harder. For example, a peak of 20 kW is punished with a higher negative reward than a peak of 10 kW.
- **2** - This experiment variant is based on *1*, but the calculation of the deviation reward is changed slightly. Instead of treating the negative and positive reward equally, the calculation for the positive reward is adjusted. This is done for the purpose of highlighting the low grid loads more.

Besides the two different variants (*1* and *2*), the experiment is again spitted into two variants. This is done in order to evaluate the impact of separate PV systems in the environment state more in detail.

In further usage, the experiment *1a* refers to using a joined PV state and variant 1. *1b* and *2b* refer to experiments of variant 1 or 2, by using the extension of separate PV systems.

### 4.1.6 Verification and Validation

The best experiment result is used for the verification and validation of the RL optimization. T ensure that the training data is different than the verification data, the generation of load profiles for the buildings is necessary. As base for the generation, the load profiles of SimBench [36] are used. Depending on the building type, matching profiles are chosen and then scaled to approx. the same yearly demand. Besides changing the load profiles, also the weather data is changed. However, there is no generation necessary, as the data is available from the cFlex project. The weather data is used for the prediction of the PV systems and heat pumps.

After the training process, the RL optimization is switched from training to evaluation mode and the trained model is used to hand over to the optimization. The data for running the simulation with no and mixed integer linear optimization is used to run the evaluation, as this makes possible to compare the results of all three optimization cases directly.

## 4.2 Result of optimization cases

In this section the results of the experiments as well as the no and linear optimization results are introduced. All simulations are made with the same settlement and same weather data. In section 4.1.1 the used settlement is already explained. The settlement contains 20 households, differentiated by load profiles, where ten households own a heat pump and four a PV installation. Besides the private buildings, five commercial and two communal buildings are part of the settlement as well as a community battery.

In some figures, the results for several simulation results are combined. To simplify the references for each result a unique identifier is introduced and listed in table 4.2.

The focus of the results lies on the key performance indicators (KPIs), the graph of the transformer power and the distribution of the power and power difference on the transformer. For the RL optimizations a plot of the total reward over the whole learning process is added. The following should give a brief overview to the different result representations to support a better understanding.

**KPIs** - The possibility of comparing different optimization results is achieved by creating KPIs. These are predefined values, which are recorded during the simulation or calculated based on simula-

| Optimization Case | Unique Identifier |
|---|---|
| No Optimization | NoOpt |
| Mixed Integer Linear Optimization | LinOpt |
| RL Optimization - Baseline | Base |
| RL Optimization - Replace Normalization | Norm |
| RL Optimization - Batch Normalization | BatchN |
| RL Optimization - Separate PV Systems | SepPV |
| RL Optimization - Dependent Reward 1a | Re1a |
| RL Optimization - Dependent Reward 1b | Re1b |
| RL Optimization - Dependent Reward 2b | Re2b |
| RL Optimization - Verification and Validation | RLOpt |

Table 4.2: Overview of the optimization case and the corresponding unique identifier

tion values afterwards. The KPIs used in this thesis are already defined for the cFlex project. A few KPIs are derived from Nikolaos et. al. [37] and referenced as *(R...)* in table 4.3. Table 4.3 gives an overview of the used KPIs and their explanation. The KPIs are divided into three groups (Power, Unitless and Energy), where the unit is specified. The values of the energy group are always related to the supply or demand loads of the EC and summed up for the entire time horizon. For a better readability the table can be found on page 36.

**Violin Plots** - To combine box plots and density graph in a visual representation, Hintze and Nelson [3] introduced 1998 the violin plot. An example of the graph is depicted in fig. 4.6. The outline of the violin-alike graph presents the distribution of the values. A wider section of the density graph indicates a higher accumulation of values, where as a narrower section indicates a lower accumulation of values. The white circle represents the median out of all values. The black bar in the middle is limited by the first and third quartile and represents the interquartile range (IQR). [3]

**Power Graph** - The power on the EC's transformer is plotted over the entire time horizon. This representation makes it possible to evaluate the appearance of peaks. The abscissa refers the time and the ordinate represents the transformer power in $kW$.

**Reward Plot** - The total reward is plotted for each time step and is represented by summing up all positive and negative step rewards. Using the curve assumptions regarding the learning process of the RL agent can be made.

For a clear arrangement the tables and figures with the results are from page 37 to page 43.

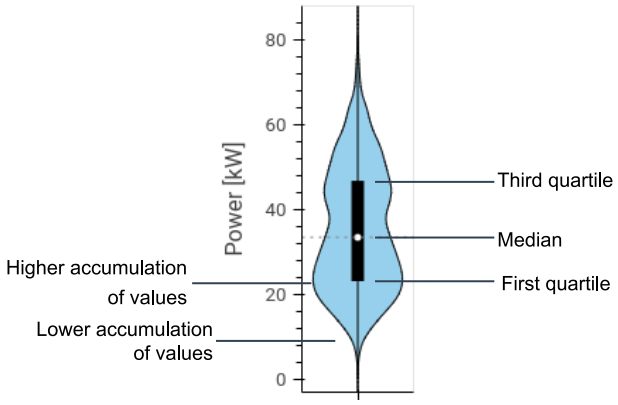



Figure 4.6: Explanation of the violin plot based on [3]

### 4.2.1 No Optimization (NoOpt)

This subsection represent the results of the EC's simulation, when no optimization is contributed. Without optimization the battery is unused, which is represented by the KPIs battery_energy_demand_sum and battery_energy_supply_sum both being zero in table 4.4. In this table, also all the other KPIs of NoOpt are listed. The plotted transformer power is visible in fig. 4.7a as blue curve. The violin plot in fig. 4.9 visualizes the distribution of transformer power. The density of values are the highest at around 24 kW and 44 kW, however at 36 kW is a distribution valley. The second violin plot for this simulation case is the distribution of transformer power difference in fig. 4.11.

### 4.2.2 Mixed Integer Linear Optimization (LinOpt)

The results of the mixed integer linear optimization developed in the cFlex project are presented in this subsection. The red line in fig. 4.7a represents the transformer power over the entire time horizon. The corresponding KPIs are listed in table 4.4. The distribution of transformer power is depicted as violin plot in fig. 4.9 and the distribution of the power difference is in fig. 4.11.

### 4.2.3 RL Optimization - Baseline (Base)

The parameter and setup for the baseline simulation of the RL optimization are introduced in section 4.1.1. For this simulation no additional improvements are done. The KPIs of this experiment can be found in table 4.4. The power difference distribution in fig. 4.11 shows a large range from around -60 kW to 50 kW between changes. In fig. 4.9 the power distribution is visualised. Whereas the transformer power is plotted in fig. 4.7b as blue line. The reward function of this RL optimization experiment can be found on page 43 (see fig. 4.13a).

### 4.2.4 RL Optimization - Replace Normalization (Norm)

In section 4.1.2 the implementation changes in the RL optimization compared to the baseline case are explained. The previous min-max-normalization for the input state is now replaced by a batch normalization at the state input of the actor and critic network. The load at the transformer is plotted as red line in fig. 4.7b. In fig. 4.13b the total reward over the time is visible. It can see that the reward goes down and then slightly up again. The violin plots are showing the transformer power distribution (see fig. 4.9) and the transformer power difference distribution (see fig. 4.11). The KPIs for this optimization are listed in table 4.4.

### 4.2.5 RL Optimization - Batch Normalization (BatchN)

For this experiment, batch normalization is added in between layers with the expecation of improving performance. More details and visualised structure of neural networks shown in section 4.1.3. The mostly falling reward graph is shown in fig. 4.13c. In fig. 4.9 the distribution of power is visualised and fig. 4.11 shows the power difference distribution on the transformer. To ease comparison reason the transformer power is in three graphs: fig. 4.7c (red), fig. 4.7d (blue) and fig. 4.7e. Concrete numbers of the KPIs are listed in table 4.4.

### 4.2.6 RL Optimization - Separate PV Systems (SepPV)

The original implementation of the environment state represents the PV systems in a EC as one variable. By aiming to achieve more accuracy the PV systems are representing themselves in the state, because the orientation and angle of irradiance is different for each PV plant. The experiment is described in more detail in section 4.1.4. In order to easily compare the result of the transformer performance, there are two diagrams showing the result: fig. 4.7d (red) and fig. 4.8a (blue). In the violin plot fig. 4.10 the power distribution is shown and in fig. 4.12 the power difference distribution. The reward function (see fig. 4.13d) stays on a equal level most of the time. The load average for SepPV is 37.56 kW and depicted with the other KPIs in table 4.5.

### 4.2.7 RL Optimization - Dependent Reward 1a (Re1a)

There are three different experiments regarding the adjustment of the reward. The basis for the reward changes for this experiment is BatchN. More about the adjustments is introduced in section 4.1.5. In this case, 7.86 kW are supplied from the EC to the grid. This KPI and the others are listed in table 4.5. Figure 4.7e shows in red the transformer power of Re1a. The distribution of power is depicted in fig. 4.10 and the distribution of power difference in fig. 4.12. The reward function is plotted in fig. 4.13e.

### 4.2.8 RL Optimization - Dependent Reward 1b (Re1b)

This experiment is based on SepPV and is one of three different experiments regarding the reward (see section 4.1.5). The KPIs are listed in table 4.5. For later comparison two diagrams are showing the transformer power: fig. 4.8a (red) and fig. 4.8b (blue). The reward function (see fig. 4.13f) almost keeps a certain level after dropping down. In fig. 4.10 the power distribution and in fig. 4.12 the power difference distribution are depicted as violin plots.

### 4.2.9 RL Optimization - Dependent Reward 2b (Re2b)

This third experiment of the dependent rewards (see section 4.1.5) is based on Re1b and therefore also have separate PV systems. The KPIs are shown in table 4.5. In two violin plots the distribution of power (see fig. 4.10) and distribution of power difference (see fig. 4.12) are shown. In fig. 4.13g the reward function is plotted and after a short drop into the negative range, the total reward increases into the higher positive range. The transformer load is shown in red in fig. 4.8b.

### 4.2.10 RL Optimization - Verification and Validation (RLOpt)

The RLOpt is done with the same implementation as Re2b. In section 4.1.6 the generation of data for the training process is explained. The model is then trained and used with the same scenario and data as the mixed integer linear optimization and all the experiments. The KPIs, transformer load graph and violin plots are captured during the evaluation phase by using the trained model. The reward function (see fig. 4.13h) is plotted during the training phase, as it is more significant. The power peak is at 94.70 kW and can be found, with all the other KPIs, in table 4.5. The load on the transformer is plotted in fig. 4.8c in blue and in fig. 4.8d in red. The power distribution is shown in fig. 4.10 and the power difference distribution in fig. 4.12.

| | KPI | Description |
|---|---|---|
| Power [kW] | transformer_power_supply_peak | Peak supply power at the EC's transformer. |
| | transformer_power_demand_peak | Peak demand power at the EC's transformer. |
| | transformer_load_average | Mean power at the EC's transformer. |
| | transformer_load_standard_deviation | Standard deviation of load at the transformer. |
| | transformer_load_interquartile_range | Range between the 25th and 75th percentiles of load values, representing the middle 50% of data. |
| | transformer_load_median | Median load value out of all EC's transformer power values over the entire time horizon. |
| | transformer_load_diff_standard_deviation | Standard deviation of power changes at the transformer. |
| | transformer_load_diff_interquartile_range | Range between the 25th and 75th percentiles of the power changes set, representing the middle 50% of data. |
| Unitless [%] | transformer_load_factor | (R1.1.2) Fluctuation of load consumption at the transformer. $mean(power)/max(power)$ |
| | average_self_consumption_rate | (R4.1.1) Amount of local generated renewable energy which is consumed locally. Averaged over the entire time horizon. |
| | average_self_sufficiency_rate | (R4.1.2) Amount of total load covered by local renewable production. Averaged over the entire time horizon. |
| Energy [MWh] | transformer_energy_demand_sum | Total demand load at the EC's transformer over the entire time horizon. |
| | transformer_energy_supply_sum | Total supply load at the EC's transformer over the entire time horizon. |
| | community_energy_demand_sum | Total demand load over the time horizon. |
| | community_energy_supply_sum | Total supply load over the time horizon. |
| | uncontrollable_energy_demand_sum | Total uncontrollable demand load over the time horizon. |
| | heatpump_energy_demand_sum | Total heat pump demand load over the time horizon. |
| | photovoltaic_energy_supply_sum | Total PV supply load over the time horizon. |
| | battery_energy_demand_sum | Total battery demand load over the time horizon. |
| | battery_energy_demand_sum | Total battery supply load over the time horizon. |

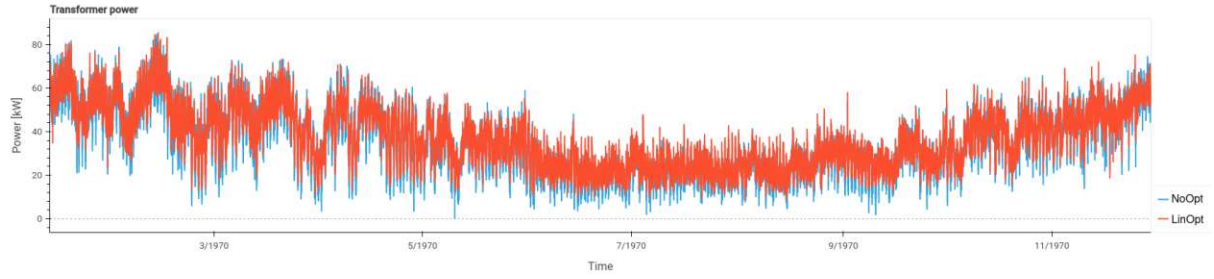
Table 4.3: Overview and Explanation of the used KPIs

| KPI | | NoOpt | LinOpt | Base | Norm | BatchN |
|---|---|---|---|---|---|---|
| Power [kW] | transformer_power_supply_peak | 0.00 | 0.00 | 0.00 | 0.00 | -3.29 |
| | transformer_power_demand_peak | 84.70 | 85.57 | 92.57 | 102.56 | 101.00 |
| | transformer_load_average | 35.25 | 36.55 | 38.53 | 38.40 | 38.55 |
| | transformer_load_standard_deviation | 14.58 | 13.12 | 13.09 | 13.24 | 13.68 |
| | transformer_load_interquartile_range | 23.26 | 20.80 | 20.25 | 20.40 | 20.37 |
| | transformer_load_median | 33.38 | 34.87 | 37.07 | 36.87 | 37.28 |
| | transformer_load_diff_standard_deviation | 3.68 | 3.55 | 5.36 | 5.59 | 7.68 |
| | transformer_load_diff_interquartile_range | 4.29 | 4.22 | 6.22 | 6.35 | 8.59 |
| Unitless [%] | transformer_load_factor | 41.6 | 42.7 | 41.6 | 37.4 | 38.2 |
| | average_self_consumption_rate | 100.0 | 100.0 | 100.0 | 100.0 | 99.5 |
| | average_self_sufficiency_rate | 10.6 | 6.0 | 0.2 | 0.6 | 0.1 |
| Energy [MWh] | transformer_energy_demand_sum | 272.42 | 282.43 | 297.77 | 296.75 | 297.95 |
| | transformer_energy_supply_sum | 0.00 | 0.00 | 0.00 | 0.00 | -0.002 |
| | community_energy_demand_sum | 291.30 | 293.34 | 292.08 | 292.55 | 299.06 |
| | community_energy_supply_sum | -25.68 | -17.74 | -1.20 | -2.69 | -8.02 |
| | uncontrollable_energy_demand_sum | 141.49 | 141.49 | 141.49 | 141.49 | 141.49 |
| | heatpump_energy_demand_sum | 149.80 | 149.80 | 149.85 | 149.85 | 149.85 |
| | photovoltaic_energy_supply_sum | -25.68 | -15.75 | -0.46 | -1.57 | -0.32 |
| | battery_energy_demand_sum | 0.00 | 2.04 | 0.74 | 1.21 | 7.72 |
| | battery_energy_supply_sum | 0.00 | -1.99 | -0.74 | -1.12 | -7.70 |

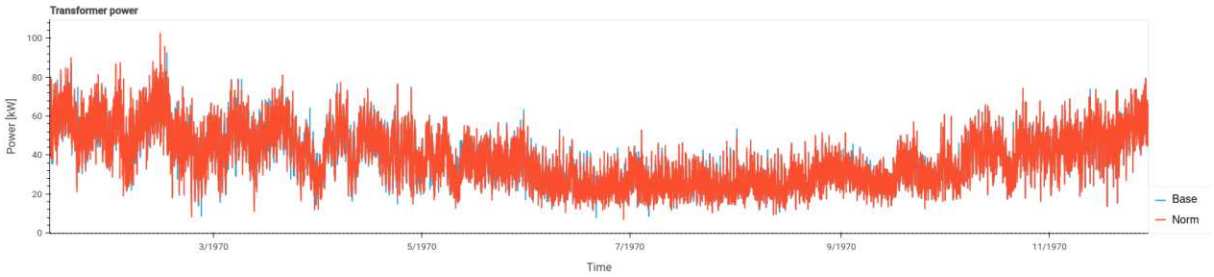
Table 4.4: KPIs for no optimization, mixed integer linear optimization and RL optimization cases: baseline, replace normalization and batch normalization

| KPI | | SepPV | Re1a | Re1b | Re2b | RLOpt |
|---|---|---|---|---|---|---|
| Power [kW] | transformer_power_supply_peak | 0.00 | -7.86 | 0.00 | -3.06 | 0.00 |
| | transformer_power_demand_peak | 95.88 | 94.57 | 94.70 | 101.60 | 94.70 |
| | transformer_load_average | 37.56 | 38.40 | 37.82 | 37.55 | 37.84 |
| | transformer_load_standard_deviation | 13.63 | 14.20 | 13.49 | 13.74 | 13.30 |
| | transformer_load_interquartile_range | 21.08 | 21.14 | 20.96 | 21.20 | 20.75 |
| | transformer_load_median | 35.55 | 37.03 | 36.02 | 35.64 | 35.92 |
| | transformer_load_diff_standard_deviation | 5.96 | 7.48 | 5.92 | 6.47 | 4.86 |
| | transformer_load_diff_interquartile_range | 6.44 | 8.29 | 6.59 | 6.93 | 5.52 |
| Unitless [%] | transformer_load_factor | 39.2 | 40.6 | 39.9 | 37.0 | 40.0 |
| | average_self_consumption_rate | 100.0 | 99.3 | 100.0 | 100.0 | 100.0 |
| | average_self_sufficiency_rate | 3.3 | 0.6 | 2.4 | 3.3 | 2.2 |
| Energy [MWh] | transformer_energy_demand_sum | 290.29 | 296.80 | 292.28 | 290.17 | 292.44 |
| | transformer_energy_supply_sum | 0.00 | -0.006 | 0.00 | -0.001 | 0.00 |
| | community_energy_demand_sum | 294.15 | 299.90 | 292.52 | 295.05 | 291.25 |
| | community_energy_supply_sum | -10.72 | -10.01 | -8.12 | -11.75 | -5.68 |
| | uncontrollable_energy_demand_sum | 141.49 | 141.49 | 141.49 | 141.49 | 141.49 |
| | heatpump_energy_demand_sum | 149.76 | 149.75 | 149.85 | 149.76 | 149.76 |
| | photovoltaic_energy_supply_sum | -7.92 | -1.41 | -5.94 | -8.03 | -5.68 |
| | battery_energy_demand_sum | 2.89 | 8.65 | 2.18 | 3.80 | 0.00 |
| | battery_energy_supply_sum | -2.80 | -8.60 | -2.18 | -3.72 | 0.00 |

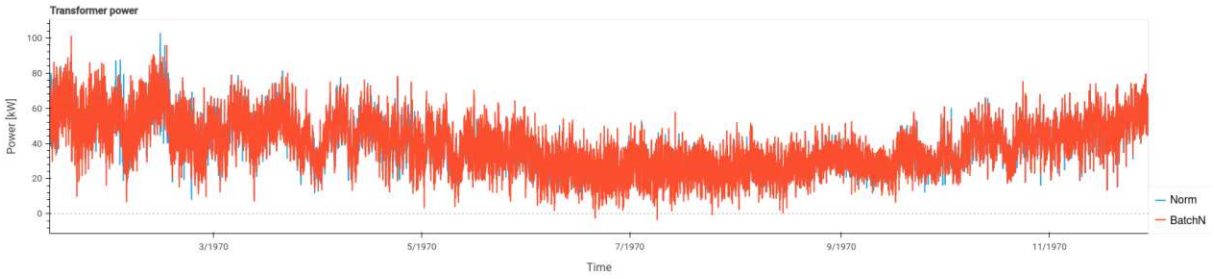
Table 4.5: KPIs for RL optimization cases: separate PV systems, dependent reward combinations and the verification



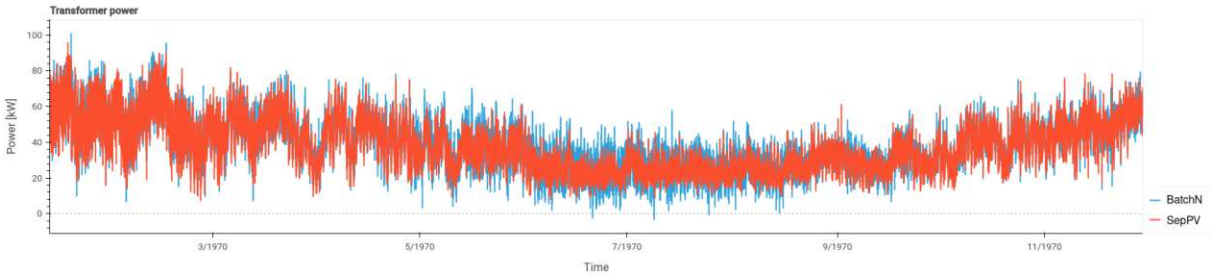

(a) NoOpt and LinOp



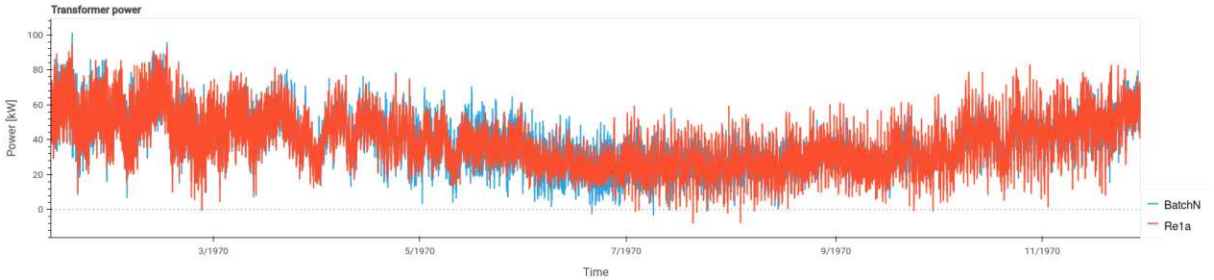

(b) Base and Norm




(c) Norm and BatchN

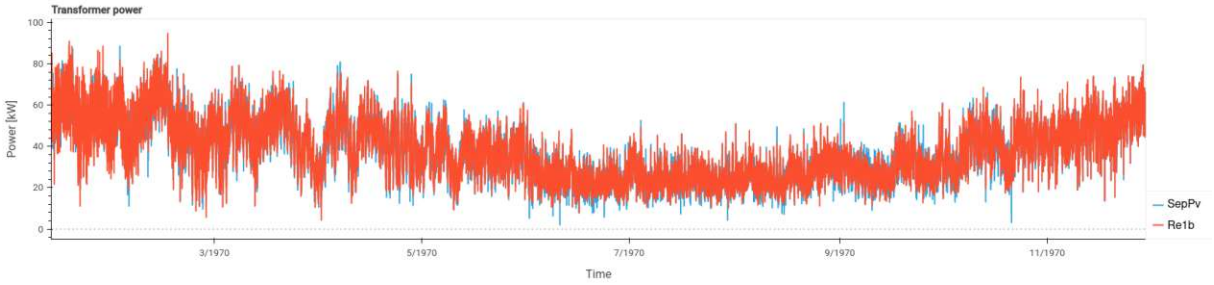



(d) BatchN and SepPV

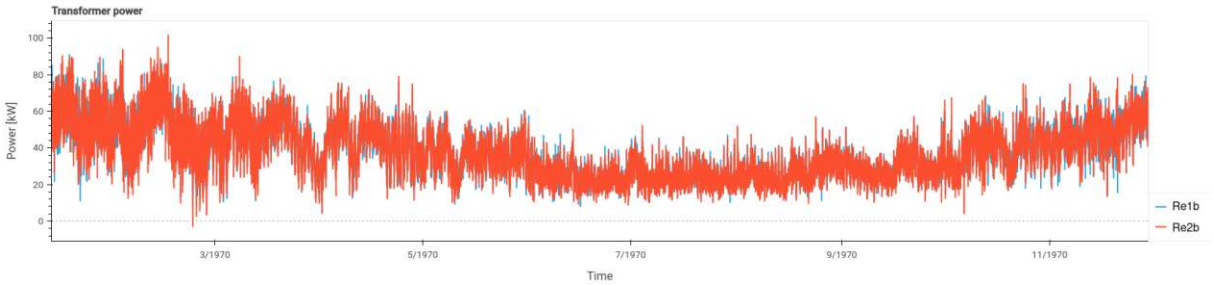



(e) BatchN and Re1a

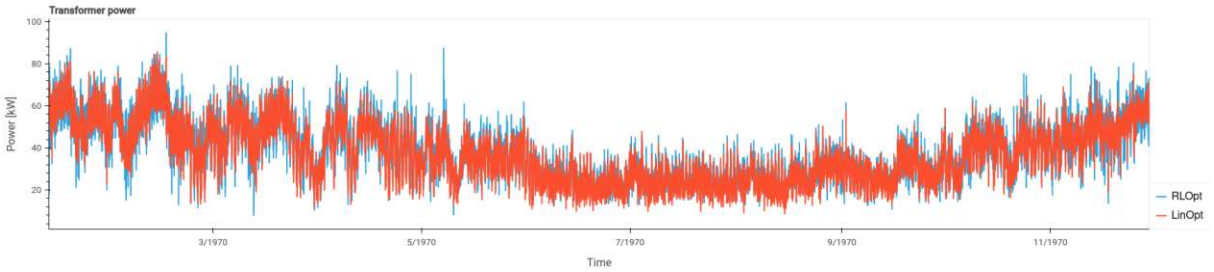
Figure 4.7: Transformer Load for NoOpt, LinOpt, Base, Norm, BatchN, SepPV and Re1a



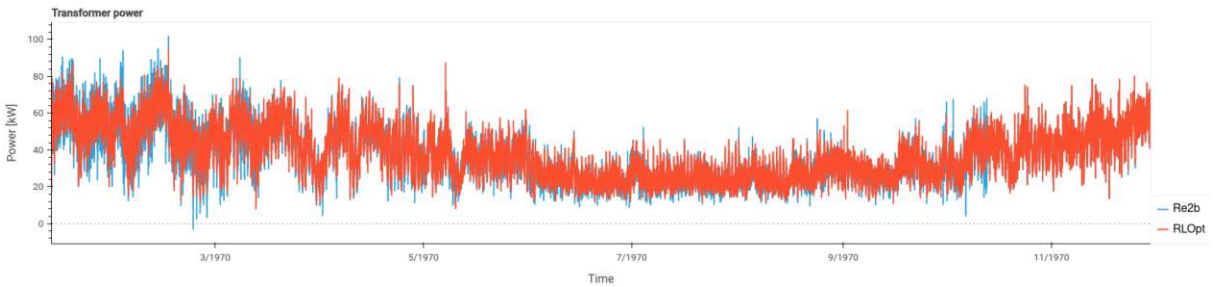

(a) SepPV and Re1b




(b) Re1b and Re2b




(c) LinOpt and RLOpt




(d) RLOpt and Re2b

Figure 4.8: Transformer Load for Re1b, Re2b and RLOpt

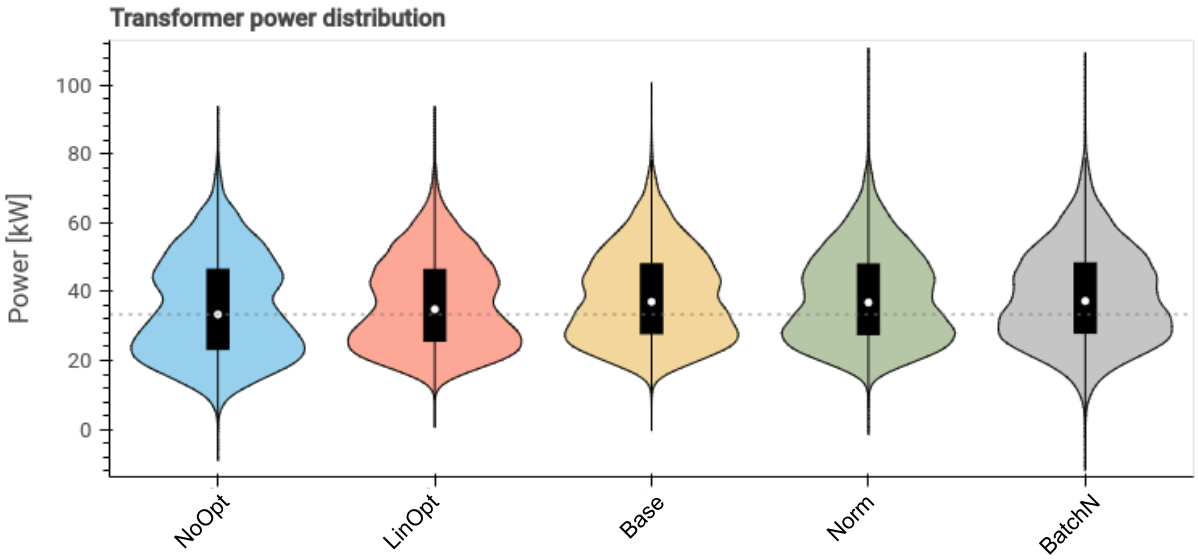



Figure 4.9: Power distribution on the transformer for no optimization, mixed integer linear optimization and RL optimization cases: baseline, replace normalization and batch normalization

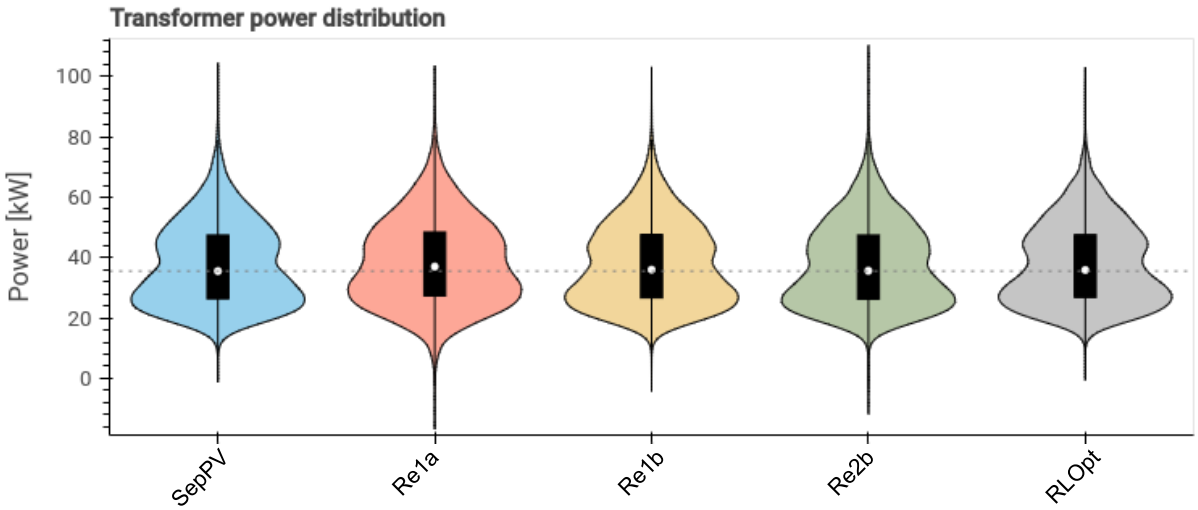



Figure 4.10: Power distribution on the transformer for RL optimization cases: separate PV systems, dependent reward combinations and verification

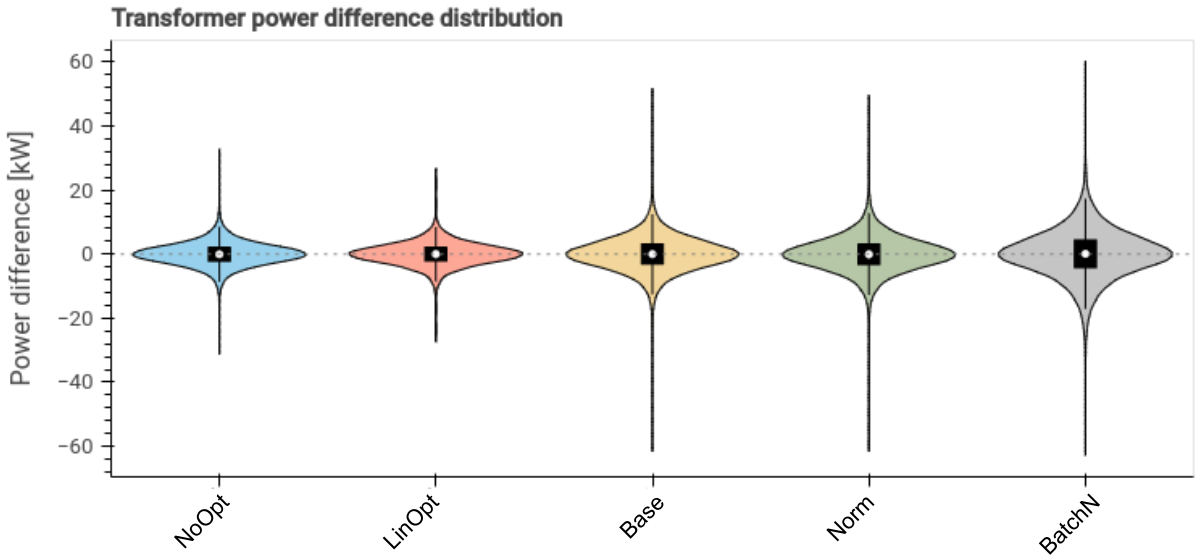



Figure 4.11: Power difference distribution on the transformer for no optimization, mixed integer linear optimization and RL optimization cases: baseline, replace normalization and batch normalization

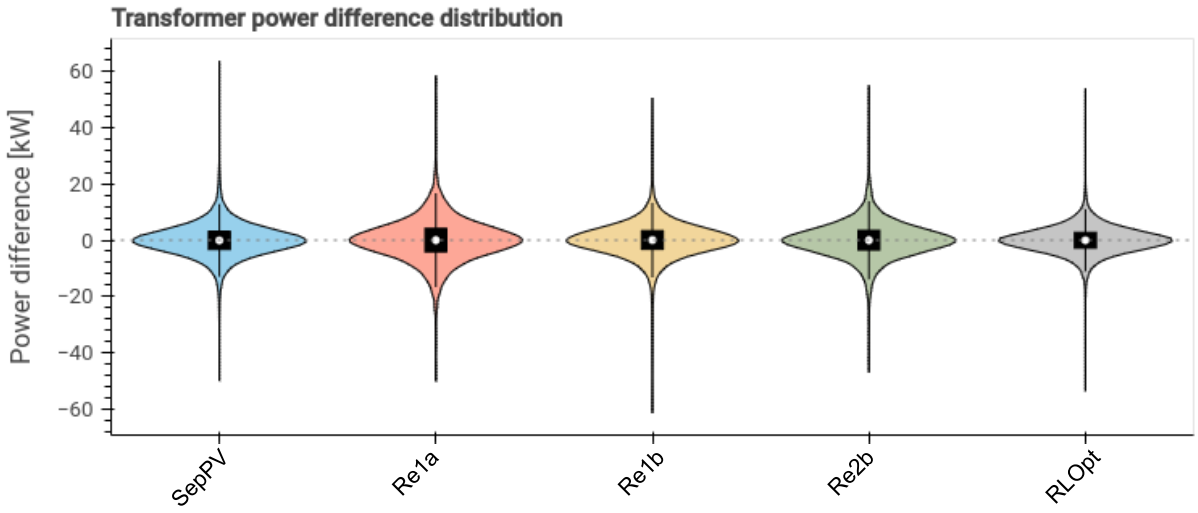
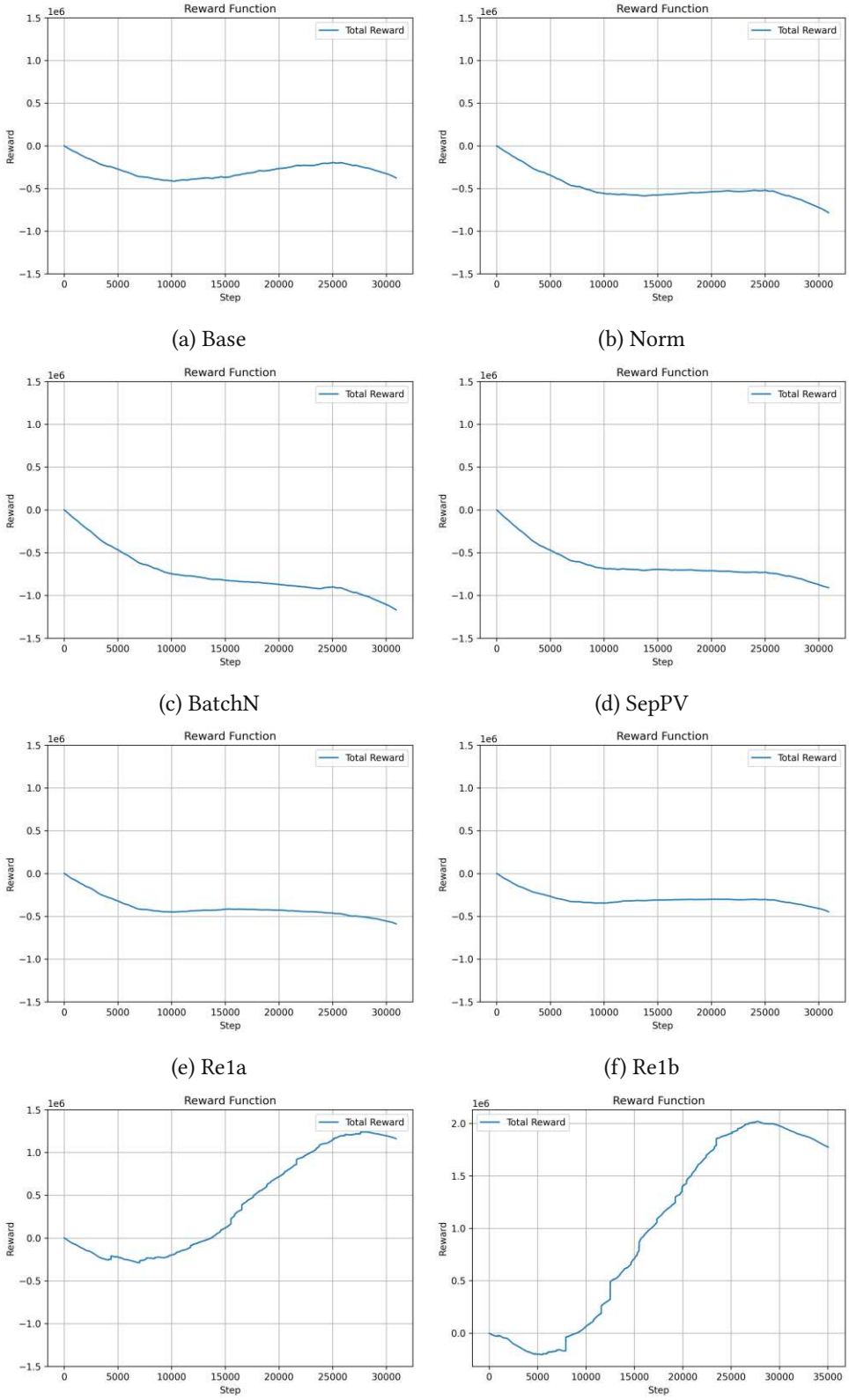



Figure 4.12: Power difference distribution on the transformer for RL optimization cases: separate PV systems, dependent reward combinations and verification




(a) Base

(b) Norm

(c) BatchN

(d) SepPV

(e) Re1a

(f) Re1b

(g) Re2b

(h) RLOpt: Training process

Figure 4.13: Reward Functions of RL optimization

# Chapter 5

# Discussion

The results from chapter 4 are discussed in this chapter. To provide a fundament for comparing the results, the used characteristics are presented. The discussion of the experiment results and the validation results are considered separately. The results of the experiments are captured during the learning process and can be compared to each other. The validation is done in evaluation mode based on the best experiment result and is compared with the mixed integer linear optimization and no optimization results.

## 5.1 Analysis of the Characteristics

The characteristics form the basis for the discussion of the results. The results, presented in section 4.2, focus on the KPIs, the transformer performance graph, and the distribution of power and power difference at the transformer. The expectations and limitations of the characteristics are presented in this section.

The KPIs can be used as an indicator of the quality and performance of the optimization. To keep the scope of this work, a few KPIs are selected for discussion. The KPIs taken into focus and their expected behavior are briefly introduced:

- **transformer_power_demand_peak** - The peak value of the power demand at the transformer is the highest peak of the entire time horizon. With regards to the goals of the optimization, reducing and smoothing the grid load, the power demand peak should be reduced.
- **transformer_load_average** - Ideally, the use of optimization does not significantly change the average load value. Due to the reduction of PV plants, it might happen that the average load increases.
- **transformer_load_diff_standard_deviation** - The transformer load makes the season visible

over the entire time horizon. The average power is higher in winter than in summer. To correct this trend, the standard deviation of the load changes are compared. A sub-goal of the optimization is to smooth the transformer's grid load. This goal can be represented by reducing the standard deviation of the power changes in transformer load.

- **transformer_load_factor** - The load factor is increased when the difference between peak load and average load is reduced. A higher load factor would mean the reduction of power peaks. However, due to the learning process in the experiments more and higher load peaks are expected at the beginning of the time horizon.
- **average_self_consumption_rate** - With the goal of reducing the grid load to a certain target value, it is assumed that the grid feed-in is also reduced. This should increase self-consumption.
- **average_self_sufficiency_rate** - Under ideal circumstances, the self-sufficiency rate would increase. Since the regulation of PV systems means a reduction in PV output power, the self-sufficiency rate may reduced.

Related to the KPIs violin plots are showing the distribution of transformer power and power difference. These violin plots containing a lot of information which are explained in more detail in section 4.2, but should be summarized for the specific use of the distribution results. Therefore fig. 4.6 is adjusted and the specific figure for the explanation of the distribution of transformer power is shown in fig. 5.1. The transformer power distribution plot represents the KPI transformer_power_demand_peak visually as the top of black vertical line. The size of the IQR is visible and comparable as the black bar in the violin plot. Whereas the white dot repesents the median value for the specific plot. The expectation by using optimization is, that the IQR gets smaller and the median value is reduced or equal compared to the un-optimized simulation. To reduce the production of energy in the PV systems can have the opposite effect. Due to the optimization, a high accumulation of values should be in lower value range. The density of high power values shall be reduced. The density is represented by outline.

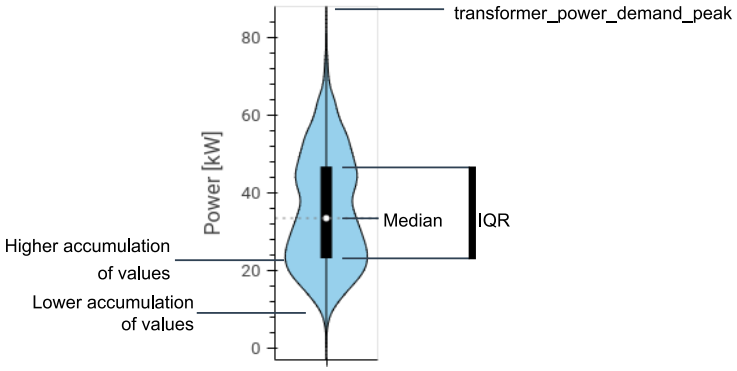



Figure 5.1: Explanation of the distribution of transformer power

The power difference is the difference between consecutive load values. The corresponding violin

plot shows the difference and the density of values. Ideally the median stays at zero, and the accumulation of values is limited to zero proximity.

The goals of the optimization are reducing and smoothing the grid load. Comparing the transformer power over the entire time horizon gives a good overview of how flat the curve is. The KPIs give total numbers, however in the experiments the RL agent is still in the training process and learns the policy. Due to that, peaks at the beginning of the load curve are more likely. To compare the graphs, the appearance of peaks after the first third of the time horizon are relevant as well as the the height of peaks. For the validation the focus shifts away from the load curve and towards the KPIs. For better comparison of the graphs, always two of them are plotted together in one coordinate system.

In RL, the reward function provides a metric to make assumptions regarding the learning process. An ideal reward curve would rise at some point, reach a certain level and stay to the level. The rising of the function is equal to the fact that the agent learns its behavior.

For a better understanding of the compared results, some of the figures from page 37 to page 43 are edited and zoomed in for specific cases.

## 5.2 Experiments

In this section the results of the experiments introduced in section 4.1 are discussed based on the characteristics of section 5.1. The sequential variants (see fig. 4.1) are compared with each other as this provides an overview of the quality of the improvements. For a better understanding the discussion of the experiments is structured in three parts: KPIs, transformer load graph and reward function.

### 5.2.1 Discussion of KPIs

In section 5.1 the KPIs to consider are explained. Based on these KPIs the results of the experiments are discussed in this section. The KPIs for RL Optimization - Baseline (Base), RL Optimization - Replace Normalization (Norm) and RL Optimization - Batch Normalization (BatchN) can be found in table 4.4, whereas RL Optimization - Separate PV Systems (SepPV), RL Optimization - Dependent Reward 1a (Re1a), RL Optimization - Dependent Reward 1b (Re1b) and RL Optimization - Dependent Reward 2b (Re2b) placed in table 4.5.

The selected KPIs to compare are in table 5.1 for all experiments to create a better overview for the discussion.

For transformer_power_demand_peak the lowest value (92.57 kWh) is at the Base experiment, whereas the highest value is 102.56 kW for Norm. In contrast to the first experiments (Norm and BatchN), the experiment with separation of the PV systems (SepPV) has a decreased peak. However,

due to the behavior of the learning process in the experiment, this value could also be a peak in the first third of the time horizon and not that meaningful. For this reason the load graphs are compared in section 5.2.2.

| KPI | | Base | Norm | BatchN | SepPV | Re1a | Re1b | Re2b |
|---|---|---|---|---|---|---|---|---|
| Power [kW] | transformer_power_demand_peak | 92.57 | 102.56 | 101.00 | 95.88 | 94.57 | 94.70 | 101.60 |
| | transformer_load_average | 38.53 | 38.40 | 38.55 | 37.56 | 38.40 | 37.82 | 37.55 |
| | transformer_load _diff_standard_deviation | 5.36 | 5.59 | 7.68 | 5.96 | 7.48 | 5.92 | 6.47 |
| Unitless [%] | transformer_load_factor | 41.6 | 37.4 | 38.2 | 39.2 | 40.6 | 39.9 | 37.0 |
| | average_self_consumption_rate | 100.0 | 100.0 | 99.5 | 100.0 | 99.3 | 100.0 | 100.0 |
| | average_self_sufficiency_rate | 0.2 | 0.6 | 0.1 | 3.3 | 0.6 | 2.4 | 3.3 |

Table 5.1: Selected KPIs for RL optimization experiments

The improvements due to the separation of the PV systems in the environment state can also be observed by the KPIs transformer_load_average, transformer_load_diff_standard_deviation and average_self_sufficiency_rate.

The transformer_load_average is lower for SepPV, Re1b and Re2b compared to the other experiment results. An explanation for this behavior can be the possibility to control the PV plants more accurate due to the separation the PV systems in the environment state. By the gain of accuracy the average load is reduced.

Transformer_load_diff_standard_deviation provides a comparison without the seasonal trend. A low value for this KPI represents a smoother grid load. The lowest value has Base with 5.36 kW. However, in the later discussion it can be seen, that this experiment has some downsides. By comparing all experiments with separate PV system handling (SepPV, Re1b and Re2b) with the hierarchical prior experiment (BatchN) it can be seen, that the separate PV systems improves the previous variant.

The average_self_sufficiency_rate is significant higher for SepPV, Re1b and Re2b, which are the only experiments with separate PV systems. This also is due to the accuracy profit at separate PV systems as previously explained.

Feed energy into the grid in combination with the total energy demand, can have an impact on the average_self_consumption_rate. This can be seen in the results for BatchN and Re1a. In Re2b energy is also fed into the grid, but the total energy demand is also lower. This affects the self consumption rate less and results in 100% like for all the experiments without a energy feed in the grid.

The transformer_load_factor is not descriptive for the experiments, due to the fact that all peaks of

the time horizon are taken into account. A low load factor is characterised by a high difference between peak load and average load. Due to the learning process during the experiments, it only makes sense to compare this KPI for the verification.

However, as the results are taken while training, the KPIs might be not so expressive. The distribution of power and power difference gives a better overview of the density trend. The distribution of power for Base, Norm and BatchN can be found in fig. 4.9, and the power difference distribution for these experiments in fig. 4.11. For SepPV, Re1a, Re1b and Re2b the power distribution is visible in fig. 4.10 and the distribution of power difference in fig. 4.12. For a better understanding of the discussion, the compared plots are summarized in fig. 5.2.

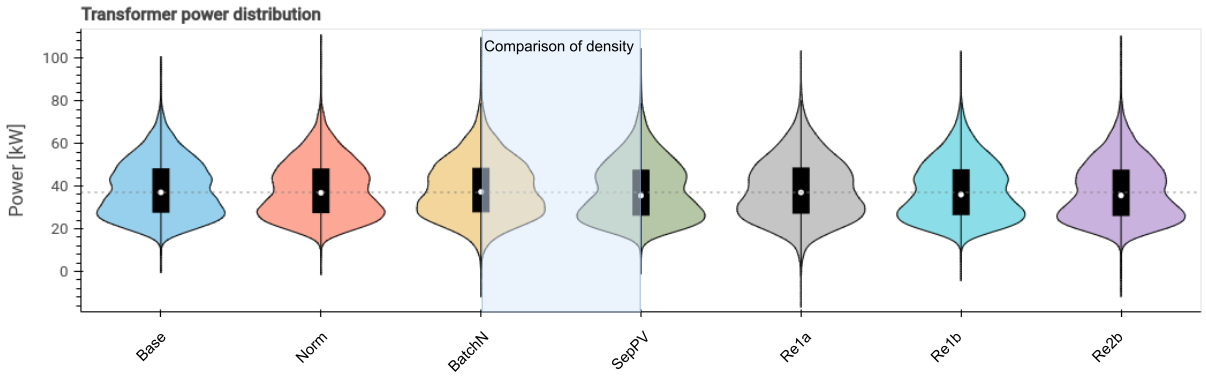



Figure 5.2: Power distribution on the transformer for discussed experiments

The light blue box in fig. 5.2, should make it easier to show the density difference between BatchN and SepPV. Since the density graph is symmetrical, it is sufficient to consider only one side. SepPV looks more like a 'Christmas tree', with a higher accumulation of lower values. Compared to this, the density of BatchN is distributed more equally over all values. The other experiments with separate PV systems (Re1b and Re2b) are also more 'Christmas tree' shaped. The conclusion from the previous discussed KPIs, that the results for the experiments with separate PV systems are better than for the other, remains valid and is supported by this observation. The white dot for each violin plot represents the median value and with the grey dotted line, the median of Base is drawn above the other results. Using this line, makes visible that the median value for SepPV, Re1b and Re2b is lower than for Base. Again, this supports the previous conclusion. The peak of each violin plot is represented by the previous discussed transformer_power_demand_peak, and therefore not considered further. Due to the learning process of the optimization in the experiments, high peaks can appear somewhere in the beginning and thereby bias the results regaring peaks. This is the reason why the power difference distribution results are not discussed for the experiments.

### 5.2.2 Discussion of Transformer Load Graphs

For the experiments, the graph of the transformer power gives a good overview about the height and amount of peaks. Due the training process only the curve of the last two thirds of the time horizon is considers. For an better comparison of the graphs, always two of them are plotted together in one coordinate system. An detailed explanation to the transformer load graphs can be found in section 5.1.

The first modification on the baseline (Base) is the replacement of the input normalization (Norm). In fig. 5.3 a transformer power graph prepared for the discussion can be seen. The amount of peaks are quite the same, but the peak height of Base (blue) are often highter than for Norm (red). A few example of these peaks are marked with black arrows.

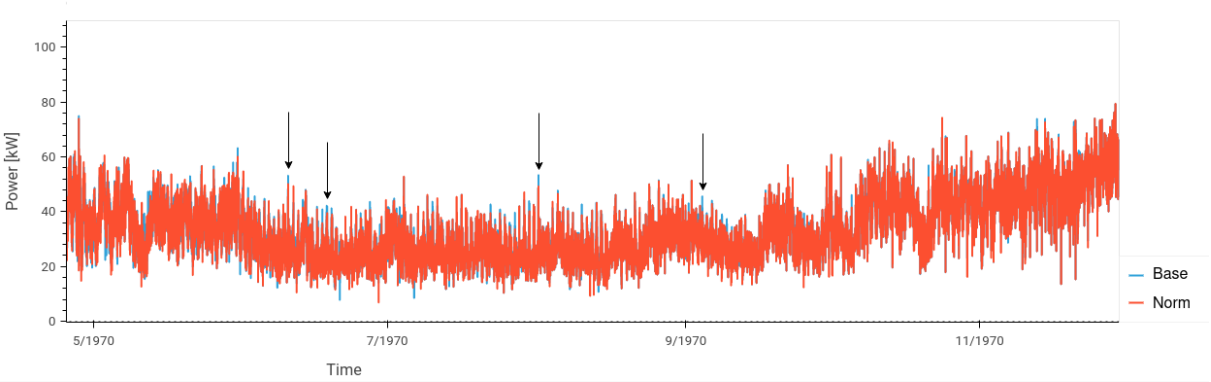



Figure 5.3: Transformer Power of Base and Norm prepared for discussion

In addition to the modified normalization (Norm), in BatchN the neural networks are adjusted with additional batch normalization in between the layers. To provide a better visualisation of the compared result of the transformer power graph in fig. 5.4 some relevant areas are surrounded by a orange box. For BatchN (red) the appearance of peaks is slightly higher, whereas the height seems to be similar to Norm (blue).

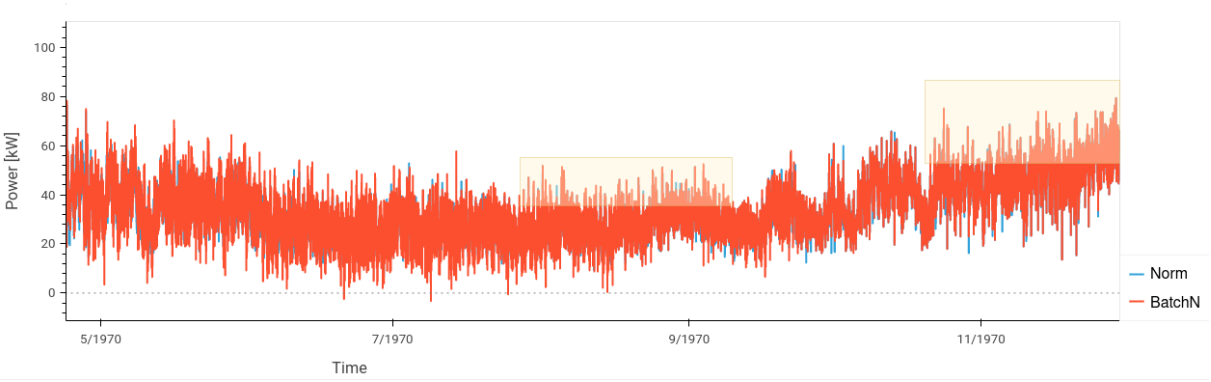



Figure 5.4: Transformer Power of Norm and BatchN prepared for discussion

However, by separating the PV state into a representation of each PV plant on its own, the graph

of the transformer load is improved. In fig. 5.5 the graphs are prepared with a orange box and black arrows, to support the readability of the graphs. The amount of peaks of SepPV are a lot less (orange box) and also the height of the peaks is reduced (black arrows).

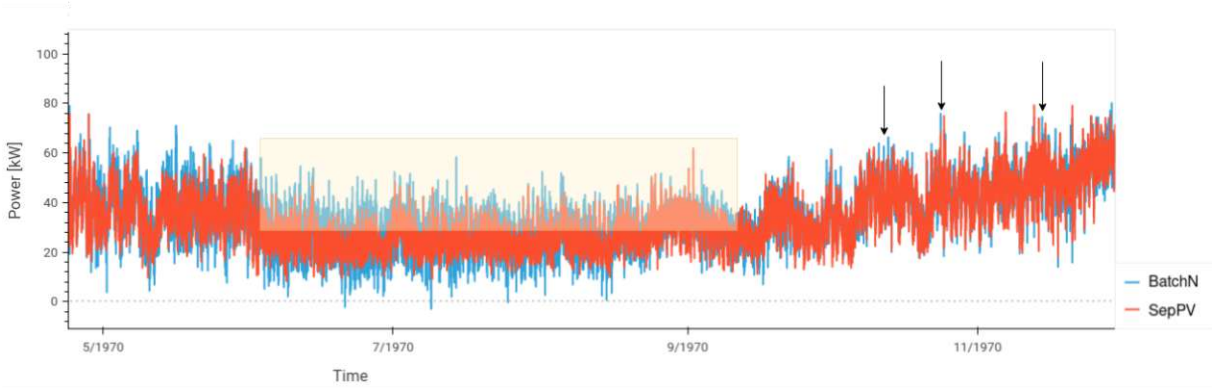



Figure 5.5: Transformer Power of BatchN and SepPV prepared for discussion

The graph for the comparison of BatchN and Re1a can be found on page 39 in fig. 4.7e. In the first half of the time horizon Re1a outperforms BatchN. However, in the second half the appearance of peaks is increased. This indicates that adding batch normalization to the neural network is not a single solution, improvements on the environment are also necessary.

One improvement on the environment side is the separation of the PV systems. Applying the reward changes of Re1a to the adjusted state representation results in the experiment Re1b. By comparing Re1b with SepPV it is visible that the appearance of peaks as well as the height of peaks are reduced in most cases. In fig. 5.6 some cases are marked with black arrows.

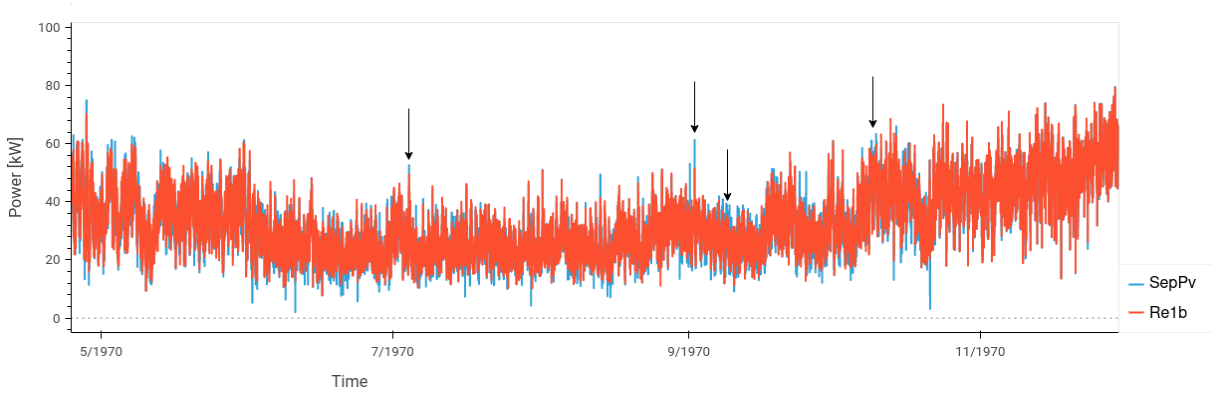



Figure 5.6: Transformer Power of SepPV and Re1b prepared for discussion

In the last experiment (Re2b) the reward is further adjusted. In fig. 5.7 various peaks are marked with arrows. It can be seen, that the height of the peaks are reduced, due to the changes done for Re2b. Especially at the end of the time horizon are less peaks than for Re1b.

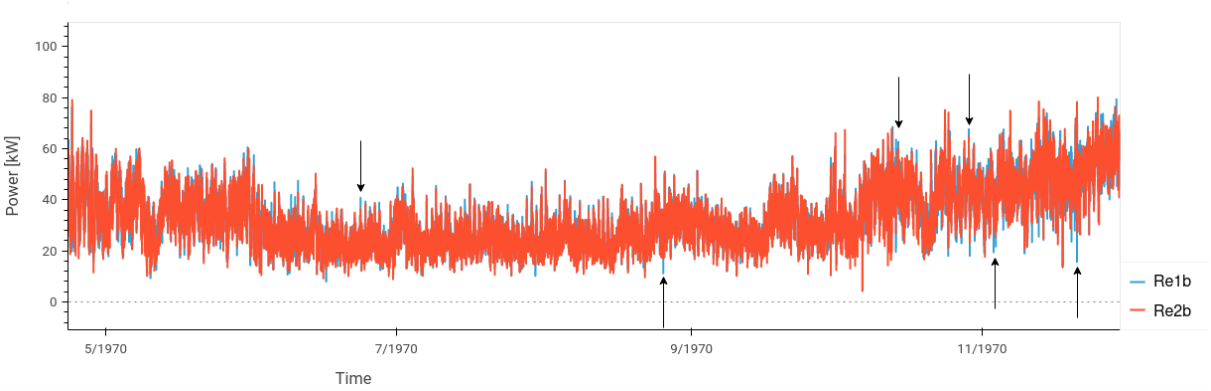
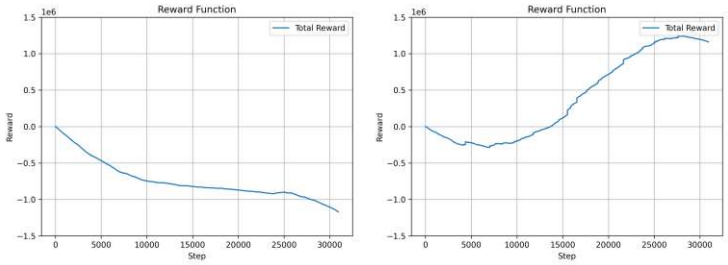



Figure 5.7: Transformer Power of Re1b and Re2b prepared for discussion

### 5.2.3 Discussion of Reward Functions

The learning behavior of the RL agent can be assessed by the plots of the reward function (see section 5.1). The time horizon is represented as steps in these reward plots, where every step is 15 minutes of the simulation. The different plots can be compared easily as the y-axis scale is the same for most of them.

The worst and best reward functions of all experiments in fig. 5.8. BatchN is the worst, because the total reward is mostly falling down. This is especially noticeable when comparing with Re2b. The reward function of Re2b is the only where the total reward reaches positive values.



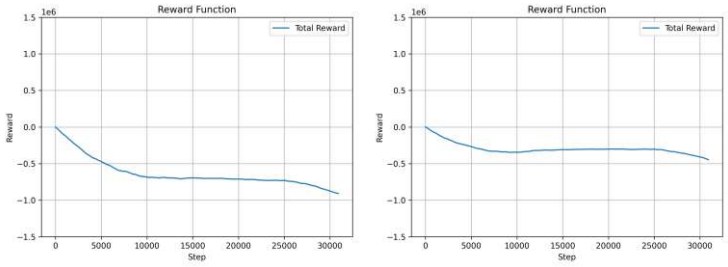

(a) BatchN (b) Re2b

Figure 5.8: Reward functions, worst and best case

There are two experiments, where the total reward is after a drop quite equal (see fig. 5.9). The learning process does not seem to be good for SepPV and Re1b, but the discussion of KPIs and transformer load graphs are still showing an improvement for both cases. Besides that, both experiments lay the basis for Re2b, where the reward function is outstanding and also the other results are good.

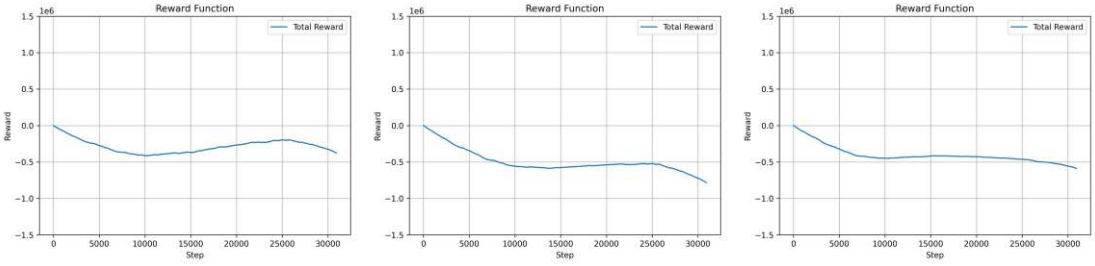
The reward plots for Base, Norm and Re1a are showing that the agent is learning (see fig. 5.10). The learning process is visualised by a rising total reward, and this happens in all three cases. After a falling curve, the total reward is rising. Not as clear and high as for Re2b, but still visible.

In section 5.2.2 it is already described that the optimization cannot only be improved by adjusting




(a) SepPV (b) Re1b

Figure 5.9: Reward functions with equal level




(a) Base (b) Norm (c) Re1a

Figure 5.10: Reward functions with light learning process

the agent, but also the environment. With regards to the reward function, only separating the PV systems does not have a good result. However, additional reward changes to make the reward more dependant from the grid load and peaks, results in a good reward function.

## 5.3 Discussion of RL optimization

In this section the results of the verification simulation with the RL optimization are discussed. For that reason the results are compared with NoOpt and LinOpt as well as with Re2b, as this was the best experiment result. For the training data, the load profiles of SimBench [36] where used and scaled to a approx. same yearly demand (see section 4.1.6). Nevertheless, comparing the 10.5 month demand of the training data with the evaluation demand (292.44 MWh, see table 4.5), the demand of the training data is approx. 30 MWh higher. This may happen due to the fact, that the seasonality was not taken into account for scaling the data. Besides that, the model learns in the training process to avoid the usage of the battery and so the battery is not used in the verification. This behavior in the training could also be triggered by the load profiles as the uncontrollable load are quite higher than for the evaluation data.

Like for the discussion of the experiments, the results are prepared for the discussion to assist understanding. The complete results can be found in chapter 4. The result of RLOpt should be first compared to the NoOpt and LinOpt cases. Afterwards a short comparison with Re2b is done, as RLOpt

is based on this experiment.

The transformer load of RLOpt has more and higher peaks than the mixed integer linear optimization (LinOpt). In fig. 5.11 a area of more peaks in visualised with a orange box, and with arrows some peaks, which are higher for RLOpt are marked.

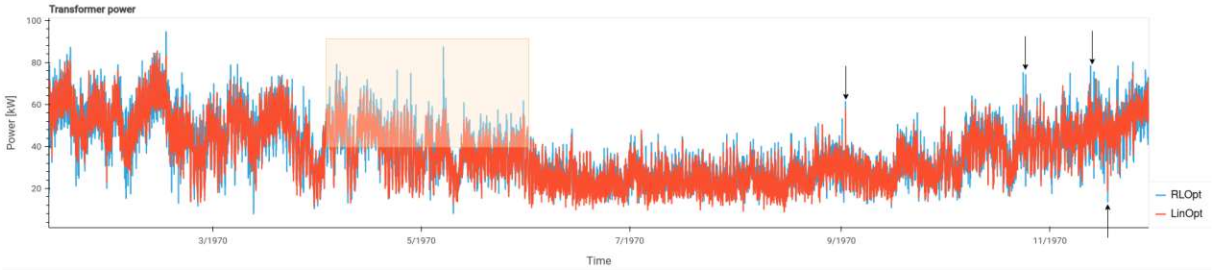



Figure 5.11: Transformer Power of RLOpt and LinOpt prepared for discussion

Comparing the transformer power graphs LinOpt is not necessary better than NoOpt. Some of the peaks, which are present for LinOpt but not NoOpt are marked with black arrows in fig. 5.12. For RLOpt, the conclusion is that this optimization does not outperform neither LinOpt nor NoOpt when comparing the transformer load diagrams.

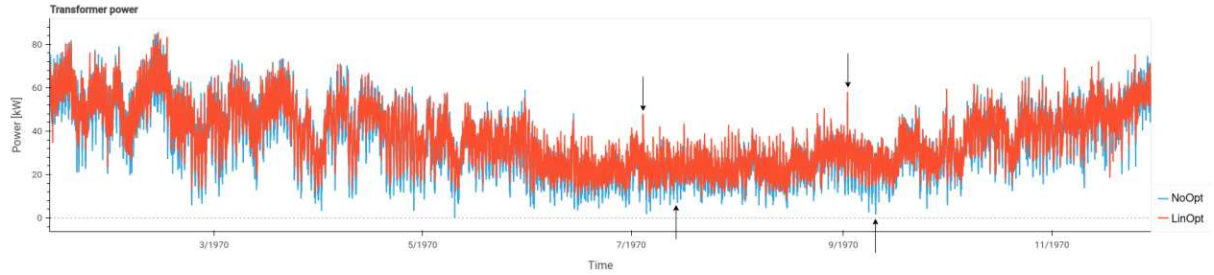



Figure 5.12: Transformer Power of NoOpt and LinOpt prepared for discussion

| KPI | | NoOpt | LinOpt | RLOpt | Re2b |
|---|---|---|---|---|---|
| Power [kW] | transformer_power_demand_peak | 84.70 | 85.57 | 94.70 | 101.60 |
| | transformer_load_average | 35.25 | 36.55 | 37.84 | 37.55 |
| | transformer_load_diff_standard_deviation | 3.68 | 3.55 | 4.86 | 6.47 |
| Unitless [%] | transformer_load_factor | 41.6 | 42.7 | 40.0 | 37.0 |
| | average_self_consumption_rate | 100.0 | 100.0 | 100.0 | 100.0 |
| | average_self_sufficiency_rate | 10.6 | 6.0 | 2.2 | 3.3 |

Table 5.2: Selected KPIs for RL optimization validation

The KPIs to consider are listed int table 5.2. Comparing the results for RLOpt and LinOpt, the mixed integer linear optimization is still better than the RL optimization. However, the gap between the values

is not that high and can probably further reduced with more modification at the RL optimization.

As RLOpt is based on the experiment Re2b, these results are compared to see possible improvements due to the separate training process.

The KPI transformer_power_demand_peak is lower for RLOpt (94.70 kW) then for Re2b (101.60 kW) (see table 5.2). The violin plots of the power distribution visualise the lower demand and supply quite well. Figure 5.13 shows the prepared plot, where the green line highlights the power demand and supply peak on the transformer. The lower peaks for RLOpt indicate that distinction between learning and evaluation phase have an effect on the height of the peaks at the transformer.

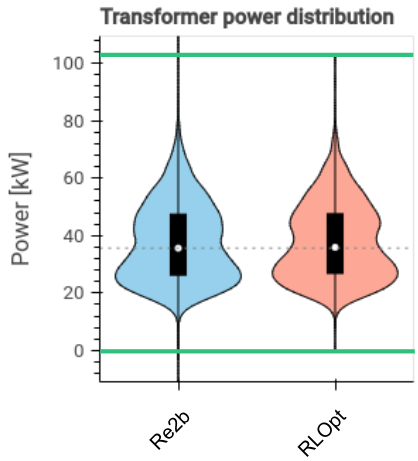



Figure 5.13: Power distribution on the transformer of RLOpt and Re2b

The load graph over the entire time horizon is plotted in fig. 5.14. The orange box indicated the first third of the time horizon. Comparing RLOpt and Re2b supports the assumption in section 5.1, that the transformer power in the experiments is not comparable in the first third of the time horizon, due to the behavior of the learning process. Re2b shows are lot more peaks in this area than RLOpt. In the remaining part of the graph, the peaks for RLOpt are less height than for Re2b, which is highlighted with black arrows.

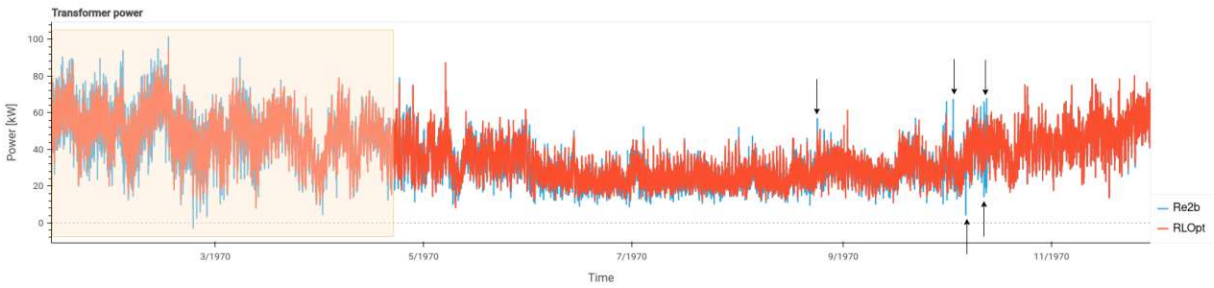



Figure 5.14: Transformer Power of RLOpt and Re2b prepared for discussion

# Chapter 6

# Conclusion and Outlook

This chapter summarizes the work and briefly discusses the results in terms of the research questions. An outlook on further steps will be given.

## 6.1 Conclusion

In this section, a short recap of the results and discussions of the RL optimization with regards to the research questions are given.

'Can the grid load be minimized and the peaks in the load smoothed by optimizing an EC with a RL algorithm?' is the main research question, which can be considered together with 'Can the minimized load and the smoothed load peaks of the RL optimization be compared with an already implemented mixed integer linear optimization?', as both are related. In section 5.3, the comparison between the linear and RL approach is discussed. The used configuration for RL optimization is behind the mixed integer linear optimization. However, the RL approach is already working, and tries to optimize the EC. Some further improvements are presented in section 6.2.

In the literature review of RL methods in the energy sector (see section 2.3), different algorithm are introduced with with their respective advantages and disadvantages. Based on the outlined methods in section 3.3, the decision for the chosen method explained. This satisfies the question: 'Which RL variant might solve the optimization problem best?'

The remaining research question is 'Are there further improvements for the RL method in order to get closer to the optimization goal of minimizing the grid load and smoothing the peaks in the load?'. The short answer is yes. The long one, can be found in different sections. In section 4.1 some improvements are described and the results of these experiments are later discussed. In this discussion (see section 5.2) it can be seen that the implementation for Base can be further improved by modifying the environment and/or agent. The amount of all possible improvements would go beyond the scope

of this work, therefore further suggestions can be found in section 6.2.

Presuming the results of the RL optimization are closer to the mixed integer linear optimization, the following could be assumed based on further improvements. The optimization is currently only used in a simulation. The results of the experiments suggest an easy usage of the RL optimization within a real EC without an extended learning phase before hand, as positive optimization results are visible after the first third. This could simplify the transition from simulation to real practice, as the barrier for EC members to use it is lowered when no year-long training process is necessary.

## 6.2 Outlook

The possibilities of algorithms to be implemented and experiments that can be carried out are nearly endless. In order to keep the scope of this thesis, the considered experiments and improvements are limited. Other possible improvements are outlined in this section. The RL optimization of this work can be improved by adjusting two components, environment and agent. Both aspects should be considered briefly in the following, starting with the environment.

Looking back at the results, it is obvious that an enhancement of the RL optimization is possible by adjusting the reward. Making modification to the reward function could result in a faster learning process, a smoother or lower grid load. Another way of using the reward for further improvements could be to prefer one flexibility over another. For example, due to such a prioritization it can be ensured that the battery is fully charged before the energy production of the PV system is reduced. Due to that, the EC could increase the self sufficiency rate and ensure the usage of most of the PV energy. Additionally, new assets that provide flexibility can be implemented. Depending on the asset behavior, results may differ and the load could be minimized and peaks be smoothed more.

Further improvements are also possible in relation to the agent. Here the network of the actor and critic could be further improved, for example by adding more layers as it is currently a simple network to get started with. Improving the whole RL optimization could also be possible by changing the RL method. Important to mention is that the new method necessary needs to provide the possibility of a continuous action space. A guide to integrate the new RL agent into the implementation of this thesis is provided in section 6.2.1.

Besides changes in the implementation, there are a few more possible improvements that can be done. Due to the limited time for data generation within this thesis and a lack of data availability, the results of the RL optimization (RLOpt) are not comparable with the mixed integer linear optimization. Improving this data by using more realistic data could enhance the learning process for the model and result in better results.

However, this thesis presents a working concept to use RL in combination with the EC controller concept and simulation in Bifrost.

### 6.2.1 Cookbook to exchange RL agent

In section 3.6 the interaction of the DDPG agent, the environment and the Bifrost framework is explained in more detail. Trying to keep the agent API as close as possible to the common one makes it easy to exchange the current agent with another RL agent. In the following an overview of the related code changes is given to simplify the exchange. Compared to common implementations the execution order of the whole environment and agent interaction is slightly changed. The sequence diagram is depicted in fig. 3.4.

The agent needs the following functions:

- *__init__()*: The initialization of the agent, its components and basic parameters happens in this function. If wanted, also the loading of an available model can be done.
- *remember()*: Can be used if the chosen RL algorithm works with a buffer to save the current sample consisting of state, action, reward and new state.
- *learn()*: It is the heart of the agent as the learning process is implemented here.
- *choose_action()*: Depending on the current state, the action is chosen by the agent in this function.
- *load_model()*: Loads a pre-saved model
- *save_model()*: Saves the current model

# Bibliography


[1] R. S. Sutton and A. Barto, *Reinforcement learning : an introduction*, second edition ed., ser. Adaptive computation and machine learning. Cambridge, Massachusetts London, England: The MIT Press, 2018.

[2] H. Dong, Z. Ding, and S. Zhang, Eds., *Deep Reinforcement Learning*. Springer Singapore, 2020.

[3] J. L. Hintze and R. D. Nelson, "Violin plots: A box plot-density trace synergism," *The American Statistician*, vol. 52, no. 2, pp. 181–184, 1998. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00031305.1998.10480559

[4] D. Banister, "The climate crisis and transport," *Transport Reviews*, vol. 39, no. 5, pp. 565–568, 2019. [Online]. Available: https://doi.org/10.1080/01441647.2019.1637113

[5] "Global energy review: Co2 emissions in 2021," *IEA, Paris*, 2022, last Accessed: June 02, 2023. [Online]. Available: https://www.iea.org/reports/global-energy-review-co2-emissions-in-2021-2

[6] K. Khan, C.-W. Su, and M. N. Zhu, "Examining the behaviour of energy prices to covid-19 uncertainty: A quantile on quantile approach," *Energy*, vol. 239, p. 122430, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360544221026797

[7] O. Celasun, A. Mineshima, N. Arregui, V. Mylonas, A. Ari, I. Teodoru, S. Black, K. Zhunussova, D. Iakova, and I. Parry, "Surging energy prices in europe in the aftermath of the war: How to support the vulnerable and speed up the transition away from fossil fuels," *IMF Working Papers*, vol. 2022, no. 152, p. 1, jul 2022. [Online]. Available: https://www.elibrary.imf.org/view/journals/001/2022/152/article-A001-en.xml

[8] M. S. Alam, F. S. Al-Ismail, A. Salem, and M. A. Abido, "High-level penetration of renewable energy sources into grid utility: Challenges and solutions," *IEEE Access*, vol. 8, pp. 190 277–190 299, 2020.

[9] B. P. Koirala, E. Koliou, J. Friege, R. A. Hakvoort, and P. M. Herder, "Energetic communities for community energy: A review of key issues and trends shaping integrated community

energy systems," *Renewable and Sustainable Energy Reviews*, vol. 56, pp. 722–744, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032115013477

[10] Österreichische Koordinationsstelle für Energiegemeinschaften, "Erneuerbare-energie-gemeinschaften," https://energiegemeinschaften.gv.at/erneuerbare-energie-gemeinschaften-eeg/ [Accessed: 31.07.2023].

[11] F. Rosenblatt, "Principles of neurodynamics. perceptrons and the theory of brain mechanisms," Cornell Aeronautical Lab Inc Buffalo NY, Tech. Rep., 1961.

[12] Y. Li, "Deep reinforcement learning: An overview," 2017.

[13] T. Zhan, "DL 101: Basic introduction to deep learning with its application in biomedical related fields," *Statistics in Medicine*, vol. 41, no. 26, pp. 5365–5378, aug 2022.

[14] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, nov 2017.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013.

[16] J. Wu and Q.-S. Jia, "A q-learning method for scheduling shared EVs under uncertain user demand and wind power supply," in *2018 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, aug 2018.

[17] Q. Dang, D. Wu, and B. Boulet, "A q-learning based charging scheduling scheme for electric vehicles," in *2019 IEEE Transportation Electrification Conference and Expo (ITEC)*. IEEE, jun 2019.

[18] U. ur Rehman and M. Riaz, "Real time controlling algorithm for vehicle to grid system under price uncertainties," in *2018 1st International Conference on Power, Energy and Smart Grid (ICPESG)*. IEEE, apr 2018.

[19] M. Kelker, L. Quakernack, and J. Haubrock, "Multi agent deep q-reinforcement learning for autonomous low voltage grid control," in *2021 IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*. IEEE, oct 2021.

[20] Y. Shu, W. Bi, W. Dong, and Q. Yang, "Dueling double q-learning based real-time energy dispatch in grid-connected microgrids," in *2020 19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*. IEEE, oct 2020.

[21] F. Kiaee, "Integration of electric vehicles in smart grid using deep reinforcement learning," in *2020 11th International Conference on Information and Knowledge Technology (IKT)*. IEEE, dec 2020.

[22] D. J. B. Harrold, J. Cao, and Z. Fan, "Battery control in a smart energy network using double dueling deep q-networks," in *2020 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe)*. IEEE, oct 2020.

[23] Y. Pan, W. Wang, Y. Li, F. Zhang, Y. Sun, and D. Liu, "Research on cooperation between wind farm and electric vehicle aggregator based on a3c algorithm," *IEEE Access*, vol. 9, pp. 55 155–55 164, 2021.

[24] M. H. Khooban and M. Gheisarnejad, "A novel deep reinforcement learning controller based type-II fuzzy system: Frequency regulation in microgrids," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 4, pp. 689–699, aug 2021.

[25] L. Yu, Z. Xu, X. Guan, Q. Zhao, C. Dou, and D. Yue, "Joint optimization and learning approach for smart operation of hydrogen-based building energy systems," *IEEE Transactions on Smart Grid*, vol. 14, no. 1, pp. 199–216, jan 2023.

[26] X. Wang, S. Chen, D. Yan, J. Wei, and Z. Yang, "Multi-agent deep reinforcement learning–based approach for optimization in microgrid clusters with renewable energy," in *2021 International Conference on Power System Technology (POWERCON)*. IEEE, dec 2021.

[27] R. Mosshammer, K. Diwold, A. Einfalt, J. Schwarz, and B. Zehrfeldt, *BIFROST: A Smart City Planning and Simulation Tool: Proceedings of the 2nd International Conference on Intelligent Human Systems Integration (IHSI 2019): Integrating People and Intelligent Systems, February 7-10, 2019, San Diego, California, USA*, 02 2019, pp. 217–222.

[28] T. Leopold, V. Bauer, A. Brathukin, D. Hauer, S. Wilker, G. Franzl, R. Mosshamrner, and T. Sauter, "Simulation-based methodology for optimizing energy community controllers," in *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*, 2021, pp. 1–6.

[29] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2019.

[30] A. Adams and P. Vamplew, "Encoding and decoding cyclic data," *The South Pacific Journal of Natural Science*, vol. 16, no. 01, 1998.

[31] J. . Han, *Data Mining concepts and techniques*, 3rd ed. Waltham, Mass.: Morgan Kaufmann, 2012.

[32] G. Kopp and J. L. Lean, "A new, lower value of total solar irradiance: Evidence and climate significance," *Geophysical Research Letters*, vol. 38, no. 1, 2011.

[33] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[34] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the brownian motion," *Physical review*, vol. 36, no. 5, p. 823, 1930.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[36] C. Spalthoff, D. Sarajlić, C. Kittl, S. Drauz, T. Kneiske, C. Rehtanz, and M. Braun, "Simbench: Open source time series of power load, storage and generation for the simulation of electrical distribution grids," 05 2019.

[37] N. Efkarpidis, A. Goranović, C.-W. Yang, M. Geidl, I. Herbst, S. Wilker, and T. Sauter, "A generic framework for the definition of key performance indicators for smart energy systems at different scales," *Energies*, vol. 15, no. 4, 2022. [Online]. Available: https://www.mdpi.com/1996-1073/15/4/1289