Bayesian Neural Network Based Control of a District Heating Network

Regelung eines Fernwärmenetzes mithilfe eines bayesschen neuronalen Netzwerks

Scientific Thesis for the procurance of the degree M.Sc.
from the Department of Electrical and Computer Engineering at the Technical University of Munich.

Supervised by	Univ.-Prof. Dr.-Ing. Sandra Hirche M.Sc. Alexandre Capone Chair of Information-Oriented Control
Submitted by	Conrad Helminger Randlkofen 84094 Elsendorf
Submitted on	07.06.2019

Abstract

District heating networks are complex systems that comprise heat production units and a water distribution network that serves many individual entities. The system dynamics include time delays, particularly in large networks. Furthermore, the demands of the customers vary and the consumptions may exhibit complex stochastic effects like multimodality and heteroscedasticity. This makes it difficult to efficiently control a district heating network. In this thesis, a control policy for a benchmark district heating system is optimized with the help of a Bayesian neural network that is trained to approximate the system dynamics.

The derived control policy performs similarly to a controller that is applied to the system to generate the dataset for training the Bayesian neural network. A slightly better performance with regards to a cost function, which is in line with desirable operation criteria of a district heating network, is obtained. The procedure of deriving a control policy by interacting with a learned model that approximates the system is proven successful. It is shown that the Bayesian neural network models complex stochastic effects that are present in the consumption behavior of the attached entities. While the essential relationships between control inputs and the system state are modelled by the Bayesian neural network, the system approximation reveals inaccuracies in state spaces that are not present in the training dataset. In conclusion, Bayesian neural network based policy search is suitable for refining an existing control policy. With this procedure, the control policy of a deployed system can be optimized while keeping the system in operation.

Zusammenfassung

Ein Fernwärmenetz ist ein komplexes System, das Wärmeenergie von Kraftwerken über ein Verteilernetzwerk zu Verbrauchern überführt. Dieses mitunter große Netz führt zu langen Laufzeiten. Weiterhin variieren die an das Netz angeschlossenen Verbraucher unter Umständen ihr Konsumverhalten, was sich in stochastischen Effekten wie Multimodalität und Heteroskedastizität äußert. Dies erschwert den Entwurf eines effizienten Reglers für ein Fernwärmenetz. In dieser Arbeit wird zunächst ein bayessches neuronales Netzwerk trainiert, sodass es die Dynamik eines simulierten Fernwärmenetzes abbildet. Anschließend wird mithilfe des gelernten Modells eine Regelungsstrategie optimiert.
Die resultierende Regelungsstrategie steuert das Fernwärmenetz nach einem ähnlichen Schema wie der zur Generierung des Datensatzes für das bayessche neuronale Netzwerk verwendete Regler. Die optimierte Regelungsstrategie erzielt gegenüber dem erwähnten Regler einen niedrigeren Wert einer Kostenfunktion, die ein gewünschtes Steuerungsverhalten abbildet. Weiterhin zeigt sich, dass das bayessche neuronale Netzwerk komplexe stochastische Effekte, welche sich aus dem Konsumentenverhalten ergeben, modellieren kann. Obgleich die essentiellen Zusammenhänge zwischen den Steuergrößen und der Systemantwort des Fernwärmenetzes abgebildet werden, sind Ungenauigkeiten in der Systemmodellierung des bayesschen neuronalen Netzwerks erkennbar, insbesondere in Bereichen außerhalb des Trainingsdatensatzes. Es ist zu schließen, dass sich dieses Verfahren zur Optimierung einer bestehenden Regelungsstrategie eignet. Die Regelung eines Systems kann ohne Unterbrechung seines Betriebs optimiert werden.

Contents

Chapter 1

Introduction

Computing methods that replicate principles of information processing in the brain have recently witnessed an upsurge due to the steadily rising availability of computational resources [Moo65]. A neuron-like mathematical model was first introduced in the late 1950s to emulate intelligent behavior [Ros58]. Even though overly-simplified compared to biology, the artificial neuron can approximate linear functions and solve simple classification problems. To handle more complex tasks, artificial neurons can be wired together into networks. Recently, researchers reported remarkable results using artificial neural networks in fields which not necessarily but most often require some notion of intelligence, such as computer vision and speech processing [Sch15, SAB14, KSH12]. The capability of approximating a wide variety of mathematical functions makes neural architectures also attractive for incorporating intelligence into computer agents and deriving control policies for complex systems, as demonstrated in [SHM+16, LHP+15]. A novel method for optimizing control policies, which relies on a class of artificial neural networks, is evaluated in this thesis by applying it to a district heating network.

District heating networks distribute centrally generated heat to commercial or residential customers through a system of insulated pipes. The heat is transported in the form of hot water and used to satisfy the customers' heating demands. A district heating system can provide higher efficiencies than local heat boilers and incorporate a variety of different heat generation stations into its network, ranging from cogeneration plants that burn fossil fuels or biomass to stations that rely on renewable energies, like geothermal or solar heating [BCD+19]. Commercially introduced in the 1870s, the total number of district heating systems has been estimated to be 80,000 as of 2013, with major ones existing in big cities as Moscow, Beijing, New York City, Seoul, Berlin and Stockholm [Wer17]. Particularly with regard to the transition from fossil fuels towards renewable energies, district heating systems are promising, as proven by Stockholm's multi energy system which is capable of efficiently handling renewable energies despite their intermittence [Lev17].

Next to its implementation, correctly controlling a district heating system — that is, regulating heat production and steering pipe flows — is key to efficiency. Most

generally, an efficient control would minimize the operator's costs while adhering to physical and non-physical constraints like consistently satisfying all customer demands. However, district heating systems are not trivial to control as they feature stochasticity and complex dynamical effects like slow responses to control inputs. Consider, for example, a heat production site's operator that wants to increase the current heat production as it becomes apparent that customers' demands are not satisfied sufficiently. Then the time delay between giving the control inputs and these taking full effect may be large. Insufficient supply may be further prolonged until the heated water eventually reaches the customer. On the other hand, stochasticity in the system is caused by the customers attached to a district heating network whose consumption behavior varies from day to day. Furthermore, nonlinear effects like thermal losses in pipes take place and the state space is usually large. All of this combined makes designing an efficient control policy challenging. The control policy has to foresee what impact actions will have on the system dynamics at some point in the future, handle nonlinear effects, and cope with stochasticity.

The procedure used to derive a control policy is known as model-based policy search. This approach requires a mathematical model that approximates the dynamics of the system that is to be controlled. Approximating the dynamics means predicting the next system state for a control input, given knowledge of the previous states. States in a dynamical system like a district heating network essentially comprise variables that can assume numerical values. In the machine learning and statistics literature, the task of approximating a function that gives a continuous-valued output is widely denoted as regression [DS14]. Many problems that are reported to be solved especially well by networks of artificial neurons are formulated as regression tasks. As learning the dynamics of a system is essentially a regression task as well, it seems reasonable to consider artificial neural networks as an option.
As mentioned earlier, the first artificial neuron models date back to the late 1950s [Ros58]. Nowadays, vastly available computational resources allow simulating large-scale neural networks of different architectures. In regression problems, neural networks are trained by fitting their parameters to a labeled dataset, i.e. a set of pairs consisting of input and desired output data samples. Most of today's widely used neural networks are deterministic and feature scalar parameters. However, this seems in conflict with the stochasticity that is present in many dynamical systems. In a district heating system, for example, yesterday's demand by a residential customer at 4 pm may deviate from today's demand at 4 pm. Therefore, fitting a deterministic neural network appropriately can be difficult as the scalar parameters provide no measure for the uncertainty linked to a prediction. Bayesian neural networks are a promising extension to cope with such stochasticity. Contrary to deterministic neural networks, parameters of Bayesian neural networks are characterized by probability distributions and inferred by posterior inference. Such a probabilistic model allows to obtain outputs that are well-distributed over the uncertainty that comes with the prediction.

Recent findings suggest incorporating random variables into the network input to account for unobservable variables affecting the system outcome in complex ways [DHDVU18]. By fitting the random variable and its weights, Bayesian neural networks can model multimodal and heteroscedastic data [DHLDVU17], as shown in Fig. 1.1. After fitting, the Bayesian neural network receives x-values and accurately models the distribution of the outputs y. Multimodality here refers to systems in which an identical input can generate two or more rather distinct outputs. Functions that are subject to a different amount of variability for different input intervals are called heteroscedastic. These are two key characteristics exhibited by customers served by district heating.
Using the Bayesian neural network to make predictions of the district heating system's future states, and thereby simulate the system itself, a control policy can be optimized by reinforcement learning. In a reinforcement learning setting, the control policy essentially has to steer the dynamical system such that a cost function is minimized. With the goal of minimizing the cumulative cost, the control policy can be optimized in an iterating manner. Model-based policy search approaches are favorable in settings where exploration is costly, as the entire space of possible control policies can be searched through and corresponding system rollouts approximated without having to interact with the real system itself. In addition, an existing control policy can be optimized for a deployed system while keeping the system in operation.

A procedure for optimizing control policies is applied to a district heating network in this thesis. The control approach requires a model that simulates the dynamics of the district heating network and a control policy described by a set of parameters. Due to the complexity and the stochasticity of a district heating system, the model is parameterized as a Bayesian feedforward fully connected neural network with stochastic inputs, as introduced in [DHLDVU17]. The model is trained by black-box α-divergence minimization (BB-α) [HLR+16]. The control policy is parameterized as a deterministic feedforward fully connected neural network. The control policy is optimized by interacting with the learned model. A cost function, which is in line with the control aims, is used to assign costs to model rollouts that are generated by applying the control policy. By minimizing the cost function, the control policy is optimized.

The remainder of this thesis is structured as follows. The introduction is extended by a review of related literature and information about the configuration and the dynamics of the district heating system. Subsequently, introductions to deterministic and Bayesian neural networks, and reinforcement learning, as relevant to the methods used in this study, are given in Chap. 2. Building upon these theoretical remarks, Chap. 3 details how a Bayesian neural network is trained to approximate the district heating network's dynamics and how the control policy is ultimately derived. After that, results of the control approach are presented and discussed in Chap. 4. Chapter 5 gives a conclusion to summarize the work, outline the obtained

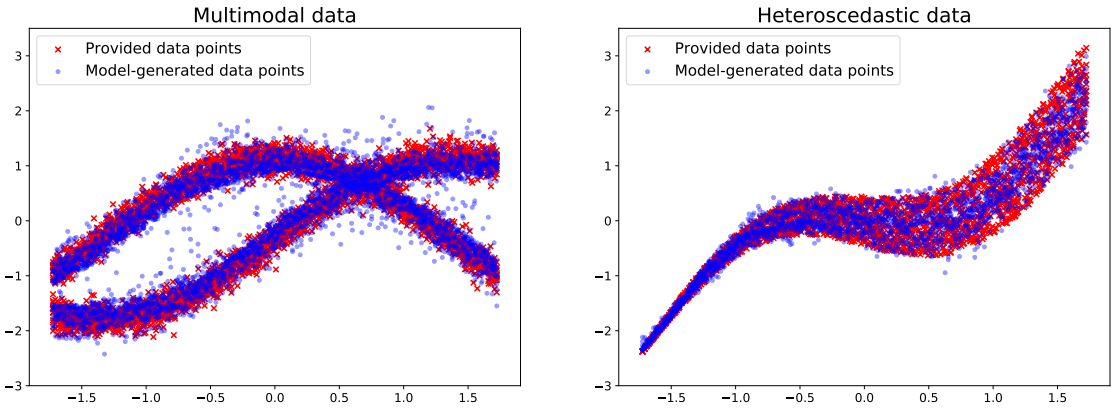


Figure 1.1: Incorporating stochastic variables in the input enables Bayesian neural networks to model multimodal and heteroscedastic data. Data points from the target distribution are depicted as red crosses, data points generated by the Bayesian neural network are depicted as blue circles. Experiments adapted from [DHLDVU17].

insights, and highlight further research directions.
In passages that mention both the benchmark district heating network and the Bayesian neural network, the former is referred to as the system while the latter is referred to as the model. The terms actions and control inputs are used synonymously in this thesis.
In mathematical formulations, lower-case symbols denote scalars, lower-case bold symbols denote vectors, and upper-case bold symbols denote matrices.

1.1 Related Work

Due to the promised savings in operational costs, optimizing the control policy of a district heating network, including the sub-field of customer load prediction, is widely studied in the literature. This section gives a brief overview of literature that is related to this work. Next to control approaches to district heating systems, a selection of control approaches to similarly high-dimensional systems is briefly given. In addition, recent advances in model-based policy search and resources regarding Bayesian neural networks are given.
It is difficult to compare different studies' results, because most authors rely on proprietary or self-implemented models for simulating a district heating network, rather than an already implemented benchmark model, to evaluate their control approach. Nevertheless, commonly used methods can be identified and qualitative remarks can be made. Sandou et al. [SFT+05] propose and simulate a complex district heating network model that takes nonlinear distribution effects into account. The controller is optimized by sequential quadratic programming [BT95], exploiting tractability of a further simplified district heating system model. The developed controller is capable of steering the dynamical system in a cost-efficient manner

and heat production peaks are observed before demand peaks. In [ISK+14], an approximate dynamic programming technique [Ber08] is applied to a district heating network model to directly optimize supply temperatures. A scalable multi-agent controller to match heat demand with the supply of thermal and electric energy is developed in [BKPW13], but not taking into account the dynamics of district heating networks. A multi-agent based operational management framework combining production, consumption and distribution aspects, while also taking market and storage into account, is given in [Joh14]. In [CVDR18], model-free policy search is applied to control a cluster of thermostatically controlled loads connected to a district heating network. As opposed to model-based policy search, the policy is here optimized by interacting with the district heating system itself instead of a learned dynamics model.
A selection of control approaches for similarly high-dimensional dynamical systems is given in the following. A model-free policy search algorithm based on deep reinforcement learning is applied to control a variety of complex systems in [LHP+15]. Control policies which compete with policies that are derived with full knowledge of the system dynamics are reported, yet this method relies heavily on exploration and is therefore rather unfavorable for physically constrained systems like a district heating network. Model predictive control is a concept widely used to control dynamical systems [FCB07]. Like model-based policy search, model predictive control relies on a model of the system, however prediction of the next states, and based on that computation of control inputs, is performed online. In recent times, variants of stochastic optimal control showed promising performance in high-dimensional control tasks due to a newly developed approach that relies on forward computation of path integrals instead of, as previously necessary, computation backwards in time [Kap07, TBS10, GTS+16]. An approach that combines model predictive control with path integral optimal control is devised in [WAT15]. A control approach for dynamical systems with big time delays is given in [SLZL16] and tested in-field on a major power plant.
Data and learning efficiency, that is, learning a system model from only a small dataset, are reported with a model-based policy search approach that relies on Gaussian processes [DR11].
Early remarks on Bayesian methods for neural networks are made in Radford's dissertation in 1996 [Nea96]. A more recent overview can be found in Gal's dissertation [Gal16].

1.2 Benchmark District Heating Network

This section provides information about the district heating network which serves as a testbed for the control approach that is evaluated. The modelling of the individual components is adapted from [SFT+04]. This framework was developed by domain

experts of the electrical utility company EDF in cooperation with scholars of the energy-oriented school Supélec. The framework features physically accurate models of pipes, pumps and heat exchangers. Mathematical descriptions of the components and general information about the assumptions made during system design are provided in this chapter. After giving the configuration of the network and information about the parameterization, special emphasis is placed on the implementation of the consumer model.

1.2.1 Component Models

The component models used to simulate the district heating network are given in this section. More detailed descriptions are given in [SFT+04] and [SFT+05].

Production Model

Production sites receive cool water from the distribution network, heat it up and return it to the network. The produced thermal power Q [W] relates to the network temperatures T [K] by

$$Q = c_{\mathrm{p}} m \big(T_{\mathrm{s}} - T_{\mathrm{r}}\big), \tag{1.1}$$

where c_{p} $[\frac{\mathrm{J}}{\mathrm{kgK}}]$ is the specific heat of water and m $[\frac{\mathrm{kg}}{\mathrm{s}}]$ is the mass flow. T_{r} and T_{s} are the water temperatures returned from the network to the producer, respectively supplied to the network from the producer. A hot water storage tank is connected in series to the production site to store thermal energy in times of low demand and draw on it in case of sudden high demands. Thermal losses during heat storage are neglected and the temperature of the water within the tank after infeed or withdrawal for a time interval $[t_0, t]$ is described as

$$T_{\mathrm{tank}}(t) = \frac{\displaystyle\int_{t_0}^{t} m_{\mathrm{in}}(\tau) T_{\mathrm{in}}\, d\tau + \rho V_{\mathrm{tank}} T_{\mathrm{tank}}}{\displaystyle\int_{t_0}^{t} m_{\mathrm{in}}(\tau)\, d\tau + \rho V_{\mathrm{tank}}}. \tag{1.2}$$

The mass flow m_{in} and the temperature T_{in} describe the water passing through the heat tank, ρ $[\frac{\mathrm{kg}}{\mathrm{m}^3}]$ denotes the relative density of water and V_{tank} $[\mathrm{m}^3]$ is the total volume of water stored in the tank. The mass flow m_{in} is regulated by a valve. The tank always returns the same amount of water that is given to it. Heat is withdrawn from the tank in case T_{in} is smaller than T_{tank}, whereas heat is saved in the tank in case the opposite holds.

Distribution Network

In [SFT+04], mechanical losses in pipes are taken into account when calculating mass flows and pressures in the network. In the scope of this thesis, mechanical losses in pipes and pumps to compensate ensuing pressure losses are neglected. Hence, mass flows and propagation time delays vary only with the opening degrees of valves in the network.

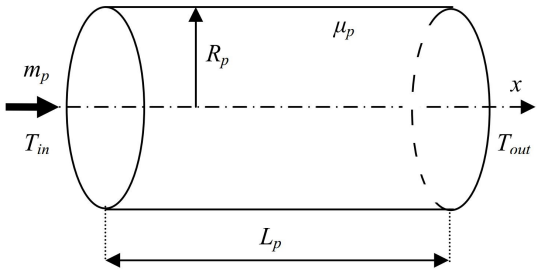


Figure 1.2: Pipe model used in the distribution network, adapted from [SFT+05].

Thermal losses due to propagation through pipes are modelled as

$$T_{\text{out}}(t) = T_{\text{ext}} + \Big(T_{\text{in}}\big(t - t_0(t)\big) - T_{\text{ext}}\Big)\, \exp\left(-\frac{2\mu_{\text{p}}}{c_{\text{p}} R_{\text{p}} \rho}\big(t - t_0(t)\big)\right) \tag{1.3}$$

with the varying time delay $t - t_0$ given by solving

$$\int_{t_0(t)}^{t} \frac{m_{\text{p}}(\tau)}{\pi R_{\text{p}}^2 \rho}\, d\tau = L_{\text{p}}. \tag{1.4}$$

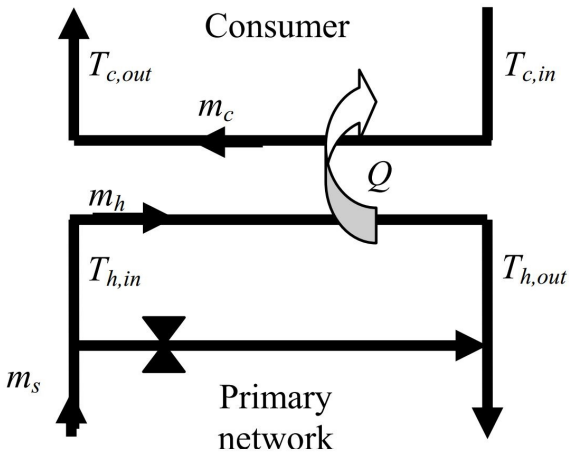


Figure 1.3: Heat exchanger model between a consumer and the distribution network, adapted from [SFT+05].

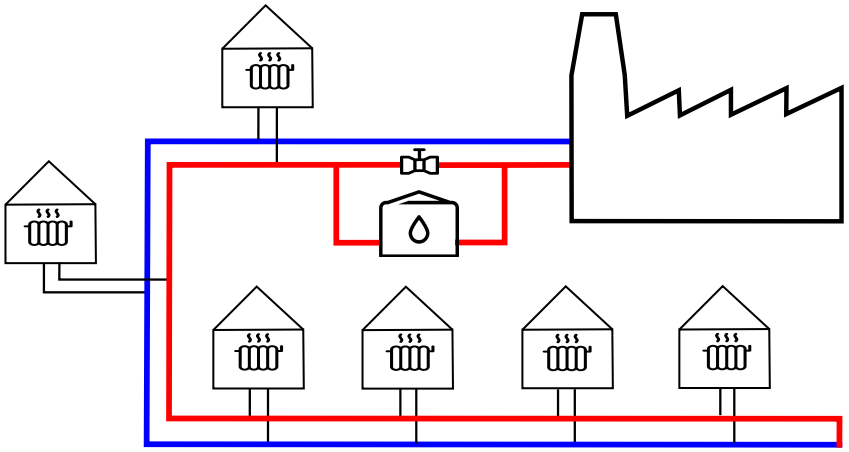


Figure 1.4: The configuration of the benchmark district heating network. The network features one production unit, one heat tank, and six customers which are connected by heat exchangers to the distribution network.

The geometric parameters are defined as depicted in Fig. 1.2. The outdoor temperature is denoted by T_{ext} and μ_{p} $[\frac{\text{J}}{\text{m}^2\text{sK}}]$ is the thermal loss coefficient of the pipe.
Consumers attached to the district heating network receive the thermal power by means of heat exchangers as depicted in Fig. 1.3.
Assuming no thermal losses during heat exchange, the thermal power given by the distribution network can be expressed as

$$Q = c_{\text{p}} m_{\text{h}} \big(T_{\text{h,in}} - T_{\text{h,out}}\big) \tag{1.5}$$

and the power received by the consumer network as

$$Q = c_{\text{p}} m_{\text{c}} \big(T_{\text{c,out}} - T_{\text{c,in}}\big). \tag{1.6}$$

The thermal energy transfer can be controlled by varying the opening degree of the valve v to regulate the mass flow m_{h} through the heat exchanger.

1.2.2 Configuration

The configuration of the benchmark district heating network is sketched in Fig. 1.4. It comprises one heat production plant, a heat tank and six consumers that are served in series. The distribution network is realized by pipes and the customers are connected to the network by heat exchangers. Thermal propagation losses and time delays that vary according to mass flows through pipes are included. Dynamics in the production site are not considered. Instead, the amount of thermal power Q to be produced for heating the water can directly be specified. The mass flow through the heat tank is regulated by a valve v. Hence, the water returned by the production site can partially be routed through the heat tank to store or withdraw

heat before distribution to the customers. Opening degrees of valves at heat exchangers, as visible in Fig. 1.3, are set constant. Thermal losses in pipes, as well as time delays during production, distribution, and in consumer cycles are taken into account. With the chosen dimensioning for the distribution network, it takes around 40 minutes for the water to cycle through the entire network. The delay that accounts for the time span in which the specified thermal power Q is generated and heats up the water in the production unit amounts to 30 minutes. Mechanical losses and pressures during water distribution are considered negligible compared to the other effects and therefore neglected. The implemented model is realized as a discrete dynamical system with a sampling period of 1 minute. The control input can be updated every 15 minutes.

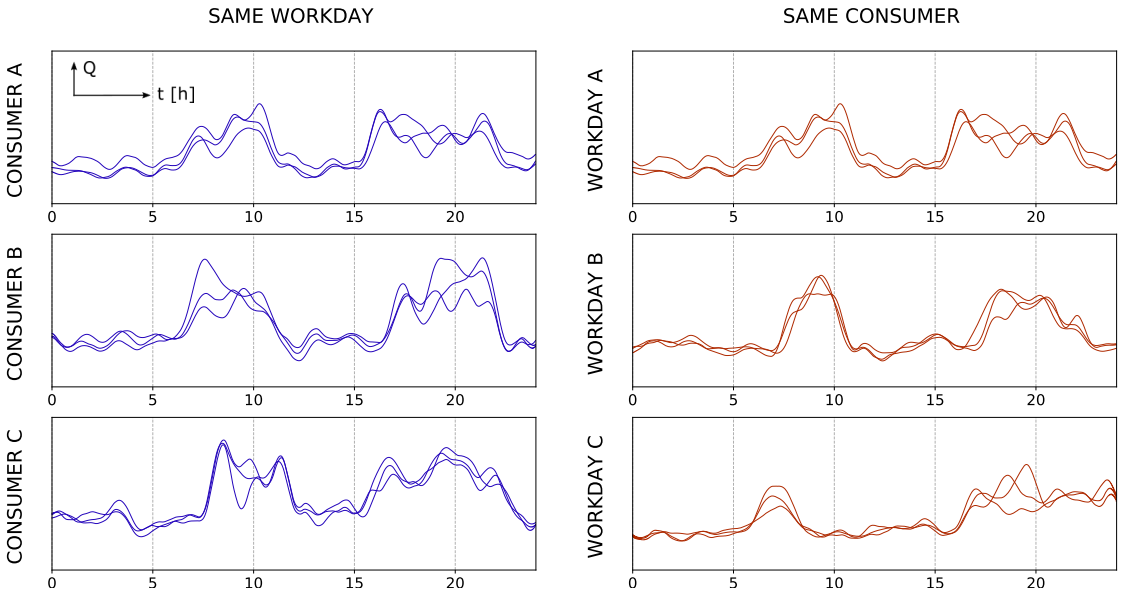
Consumer model

Consumers demand a constant water temperature at their heat exchangers. The water in the consumer cycle is cooled down according to the consumption and sent back to the heat exchanger. As a consequence, at the heat exchanger, the water in the distribution cycle is also cooled down by the amount of thermal power that the consumer received. Time delays that account for the duration that the water takes to traverse the consumer cycles are set constant according to the total volume of the consumer cycle.
Consumptions of the individual consumers connected to the network are nonlinear superpositions of multiple nested random variables, some related to time and external temperature, while some independent. The design of the consumption sampler is of particular importance, as it is the key driver of the system's stochasticity and strongly influences the difficulty of the control problem. If consumptions were entirely random, then any algorithm would fail to learn an accurate system approximation. In other words, the amount of stochasticity created by independent random variables sets an upper limit to the model's performance. To ensure a basic learnability of the system, major stochastic components of the consumption generator have to be driven by latent variables. Therefore, each consumer $c \in [0, ..., 5]$ is modelled by characteristics $\mathcal{C}_d^c$ which depend on the day of the week $d \in [0, ..., 6]$. The characteristics comprise a baseline consumption $b \in \mathbb{R}$, and scalar numbers n_s, n_m and n_l which give the amount of small, medium and large peaks. Corresponding vectors $\mathbf{t}_i \in \mathbb{N}^{n_i}$ specify the times at which the peaks appear, while $\hat{\mathbf{a}} \in \mathbb{R}^{n_i}$ specifies the peaks' amplitudes. Therefore, a set of consumer characteristics is defined as

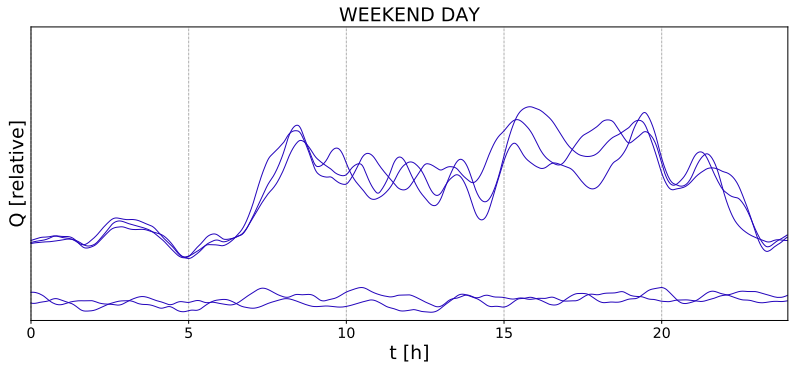
$$\mathcal{C}_d^c = \left\{ b^c, [\mathbf{t}_s^c, \hat{\mathbf{a}}_s^c], [\mathbf{t}_m^c, \hat{\mathbf{a}}_m^c], [\mathbf{t}_l^c, \hat{\mathbf{a}}_l^c] \right\}. \tag{1.7}$$

Hence, at the beginning of a system simulation, seven such characteristics are initialized for every consumer. Characteristics for workdays generally feature peak consumptions during the morning and the evening hours. Characteristics for weekend days are bimodal to account for days at which consumers are not at home. The

bimodality is implemented by sampling from a uniform binary probability distribution whether the consumer is at home.



(a) Generated consumption curves for workdays. Three curves are shown in each plot.



(b) Consumption curves of a consumer for a weekend day.

Figure 1.5: Exemplary heat consumption curves of consumers served by the district heating network. The curves are sampled from the consumers' consumption characteristics, as detailed in Sec. 1.2.2.

To obtain a sufficient amount of data for training the Bayesian neural network, a simulation is normally rolled out over several weeks. Hence, daily consumption curves are sampled drawing from the introduced characteristics $\mathcal{C}_d^c$. The baseline is adapted according to the external temperature and the peaks $\hat{\mathbf{a}}$ are disturbed by additive and multiplicative noise. The peaks are interpolated to obtain a continuous curve over 24 hours. Adding the peaks to the baseline b yields the final consumption curve. Exemplary curves for weekdays are depicted in Subfig. 1.5a. The plots on the

left show that, for the same workday, different consumers may exhibit different times and amplitudes of peak consumptions. The right plots show that the same consumer may exhibit different times and amplitudes of peak consumptions for different workdays. Considering an arbitrary plot, it can be noted that consumption curves of the same consumer for the same day are subject to variation. As the variation differs with the daytime, the consumption behavior is heteroscedastic. Consumption curves for weekend days are depicted in Subfig. 1.5b. When not at home on a weekend day, the consumption is low as a consequence. When at home, the consumption is high throughout the active part of the daytime. Hence, the consumers exhibit bimodal behavior on weekends. In addition, it can be observed that the variability of the curve is higher during daytime than at night and the early morning hours. Therefore, the consumption curves on weekends show heteroscedasticity as well.
The sampling process ensures a general learnability of the dynamics, as consumptions are calculated from latent variables which are assumed to be learnable in principle. Mixing noise into the samples adds uncertainty to the consumptions and thus raises the difficulty of the learning task.
Several control actions, observable and unobservable states during a simulation of the district heating system are visualized in Fig. 1.6. The consumptions of the individual consumers are depicted in the bottom left. The temperatures of the water returned by the customers $T_{\mathrm{c,out}}$ are depicted in the bottom right. The consumer temperature curves are essentially the mirrored consumption curves, as given in (1.6). The course of the return temperature in the distribution cycle in turn reflects the average course of the consumer temperatures. Hence, the stochastic consumers play a major role in the system dynamics.

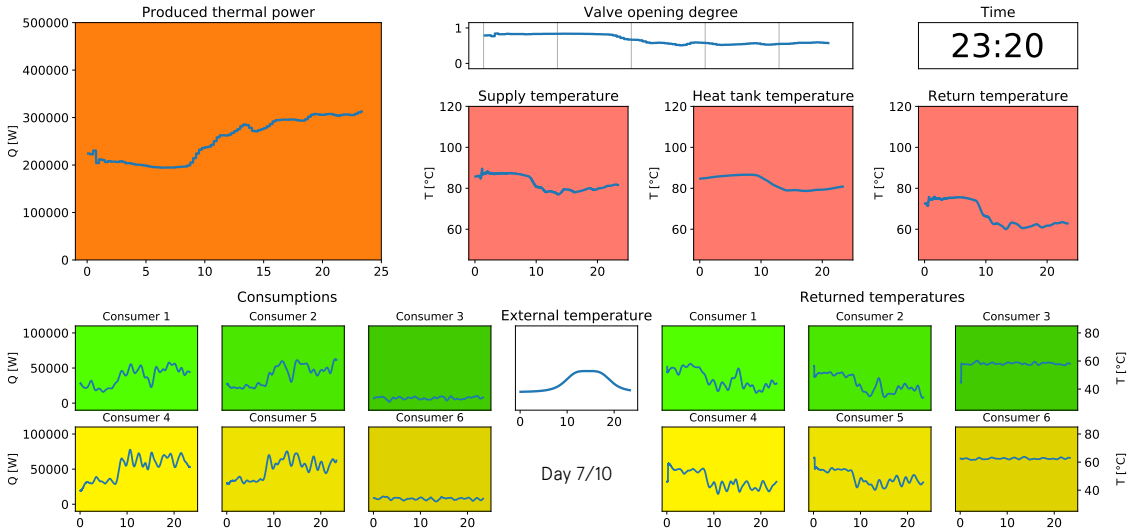


Figure 1.6: A simulation of the district heating network. To keep a clear overview, axes labelling and scaling is omitted when not subject to ambiguity.

Chapter 2

Theoretical Background

This chapter lays the theoretical foundation of the methods that are used to derive a control policy. First, the type of artificial neural networks that is used for control policy optimization is introduced in its most common form. After pointing out a connection between optimization of the introduced deterministic neural networks and Bayesian statistics, a framework of Bayesian neural networks for stochastic settings is introduced. Finally, a brief introduction to policy search is given on the basis of the reinforcement learning framework, so that the reader can understand the policy optimization algorithm explained in Chap. 3.

2.1 Neural Network Theory

Fully connected feedforward neural networks, also denoted as single- or multi-layer perceptrons, were the first type of neural networks devised [Sch15]. Such networks are hierarchically organized in layers of neurons. Full connectedness means that every neuron of one layer is connected to all neurons of the previous and subsequent layer. In a feedforward topology, the input passes unidirectionally through the network without being sent back to an already visited layer as in recurrent networks [Sch93]. As all neural architectures dealt with in this study are fully connected and feedforward, the expression fully connected feedforward may at times be omitted for simplicity.
This section explains the basic concepts of standard and Bayesian neural networks. To draw a clear distinction between these two variants, the term deterministic is used to refer to standard neural networks. The parameters of deterministic networks are scalar. In Bayesian neural networks, parameters are described as probability distributions.

2.1.1 Deterministic Neural Networks

A simple feedforward fully connected architecture is depicted in Fig. 2.1. Individual neurons are linked to each other by weighted connections. The layer of neurons

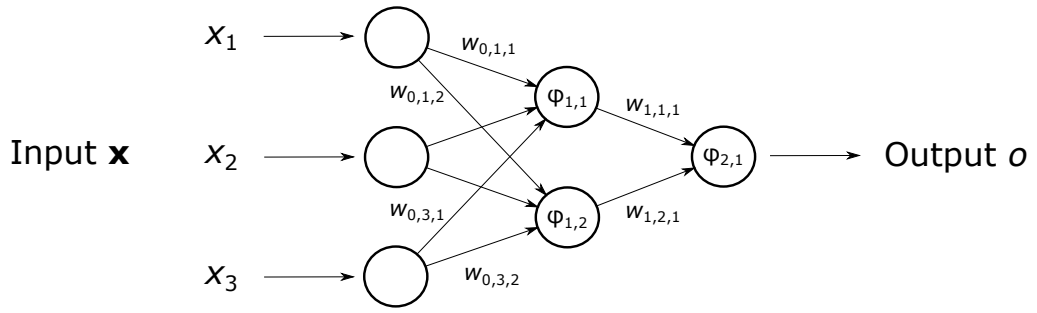


Figure 2.1: A simple fully connected feedforward neural network. According to its connectivity, the input vector $\mathbf{x}$ is transformed to output o by weights $\mathcal{W}$ and activation functions ϕ.

that pass on the input vector $\mathbf{x}$ is called the input layer, the middle layer is called a hidden layer and the last neuron constitutes the output layer. A neural network can have an arbitrary number of neurons in each layer and an arbitrary number of hidden layers. When a neuron sends its output to a following neuron, the output is multiplied by the weight w in between. Hidden and output neurons collect all weighted inputs from previous neurons and apply some, often nonlinear, activation function ϕ to the sum. Typical activation functions are sigmoid functions like the tangens hyperbolicus (tanh) and rectifiers, e.g. the rectified linear unit (ReLU) with $\phi(x) = \max(0, x)$ [XWCL15, NH10]. The output o of the neural network depicted in Fig. 2.1 is calculated as

$$o = \phi_{2,1}\left(\sum_{b=1}^{2} w_{1,b,1}\,\phi_{1,b}\left(\sum_{a=1}^{3} w_{0,a,b}\,x_a\right)\right), \tag{2.1}$$

where $w_{l,i,j}$ is the weight from neuron i in layer l to neuron j in layer $l+1$ and $\phi_{l,i}$ the activation function of neuron i in layer l. With U_l being the number of neural units in layer l, the output o of hidden neuron j in layer l is

$$o_{l,j} = \phi_{l,j}\left(\sum_{i=1}^{U_{l-1}} w_{l-1,i,j}\,o_{l-1,i}\right). \tag{2.2}$$

Hence, the output vector $\mathbf{o}$ of a generic fully connected feedforward neural network with L computation-performing layers is

$$\mathbf{o} = \boldsymbol{\phi}_{\mathrm{L}}\left(\sum_{j=1}^{U_{\mathrm{L}-1}} \mathbf{w}_{\mathrm{L}-1,j}\,\phi_{\mathrm{L}-1,j}\left(\sum_{i=1}^{U_{\mathrm{L}-2}} w_{\mathrm{L}-2,i,j}\,\phi_{\mathrm{L}-2,i}\Big(\ \ldots\ \sum_{a=1}^{U_0} w_{0,a,b}\,x_a\Big)\right)\right). \tag{2.3}$$

In practice, a bias-term is often appended to computation-performing layers. This is a constant-valued node that neither receives inputs from previous neurons nor

activates, but is fully connected through weights to the subsequent layer [Zel94]. The bias provides additional flexibility in fitting the data, as it makes it possible to shift the intersect of a neuron's activation function with the y-axis.

Training

Neural networks essentially learn by adjusting the weights between their neurons. Training can be conducted by supplying pairs of inputs $\mathbf{x}$ and desired outputs $\mathbf{y}$ to the network and fitting its weights accordingly. This means finding weights so that the neural network generates output $\mathbf{y}$ when given input $\mathbf{x}$. In case no desired outputs are available, the weights can be fitted so that the network's output minimizes an objective function tailored to the problem. In both cases, weights are most often optimized by gradient-based methods. The loss, that is, either the output of an objective function or a distance-measure between network output and desired output, is propagated backwards through the network to compute gradients with respect to each weight [Wer74]. Weights are then updated in the negative direction of the gradient to reduce the loss. To avoid local minima and speed up the optimization, parameter update algorithms like Adaptive moment estimation (Adam) can be used [KB14].

2.1.2 Bayesian Learning for Neural Networks

The most common loss measure for supervised optimization of deterministic neural networks is the mean squared error (MSE) function. Given N pairs of samples $\mathbf{x} \in \mathbb{R}^{1\text{xD}}$ and corresponding labels $y \in \mathbb{R}$, weight optimization by mean squared error is given as

$$\mathbf{w}_{\text{MSE}} = \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} (y_i - \mathbf{x}_i \mathbf{w})^2 \tag{2.4}$$

for a model with parameters $\mathbf{w} \in \mathbb{R}^{\text{D}}$ that computes as output $\mathbf{xw}$ given input $\mathbf{x}$. In other words, the task requires finding the weights $\mathbf{w}$ which minimize the MSE between desired and actual output.
From a probabilistic point of view, this corresponds to maximizing the likelihood that a model with weights $\mathbf{w}$ generates the set of labels $\mathbf{y} \in \mathbb{R}^{\text{N}}$ for the set of samples $\mathbf{X} \in \mathbb{R}^{\text{NxD}}$ [GBC16]. Consider now a probabilistic version of the model which computes its output as $\mathbf{xw} + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Then, maximum likelihood estimation (MLE) defines the optimal weights $\mathbf{w}$ as

$$\mathbf{w}_{\text{MLE}} = \arg\max_{\mathbf{w}} \log p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}). \tag{2.5}$$

To make the relation to MSE-minimization evident, the log-likelihood is formulated as

$$\log p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = -\frac{N}{2} \log 2\pi\sigma^2 - \sum_{i=1}^{N} \frac{(y_i - \mathbf{x}_i \mathbf{w})^2}{2\sigma^2} \tag{2.6}$$

by using the definition of the probability density function of a normal distribution. This procedure is based on maximizing the likelihood of the distribution of outputs $\mathbf{y}$. Hence, only the most likely set of weights $\mathbf{w}$ is returned while not taking uncertainty over this estimation into account. In settings where data is sparse or identical samples point to different labels, it would be very useful to have information about the uncertainty linked to the model's output, even perhaps obtain a distribution over possible outputs.
In contrast to MLE, maximum a posteriori estimation (MAP) returns a probability distribution of the model's weights. Here, the probability $p(\mathbf{w} \mid \mathbf{X}, \mathbf{y})$ of the weights after observing data has to be maximized. Having the set of weights $\mathbf{w}$ as a probability distribution allows to sample multiple sets of scalar weights to obtain model outputs distributed over the parameter uncertainty. With Bayes' formula, the term can be formulated as

$$p(\mathbf{w} \mid \mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathbf{X}, \mathbf{w})\, p(\mathbf{w})}{p(\mathbf{y} \mid \mathbf{X})}, \tag{2.7}$$

with the prior probability of the weights denoted as $p(\mathbf{w})$. Then, MAP defines the optimal weights as

$$\mathbf{w}_{\mathrm{MAP}} = \arg\max_{\mathbf{w}} \log p(\mathbf{w} \mid \mathbf{X}, \mathbf{y}). \tag{2.8}$$

As the denominator in (2.7) is independent of weights $\mathbf{w}$, the optimization task can be stated as

$$\mathbf{w}_{\mathrm{MAP}} = \arg\max_{\mathbf{w}} \log p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) + \log p(\mathbf{w}). \tag{2.9}$$

Hence, while MLE just considers evidence in the form of the data, MAP additionally takes prior knowledge about the weights into account. In (2.9), the prior $p(\mathbf{w})$ acts as a regularization term and may be used to prefer simpler models over more complex ones to avoid overfitting [SHK+14].
In Fig. 2.2, a simple neural network with probabilistic weights is depicted. The set of weights is described as a probability distribution, thus the model's output is also a distribution. The distribution of the weights can be fitted to data by MAP.

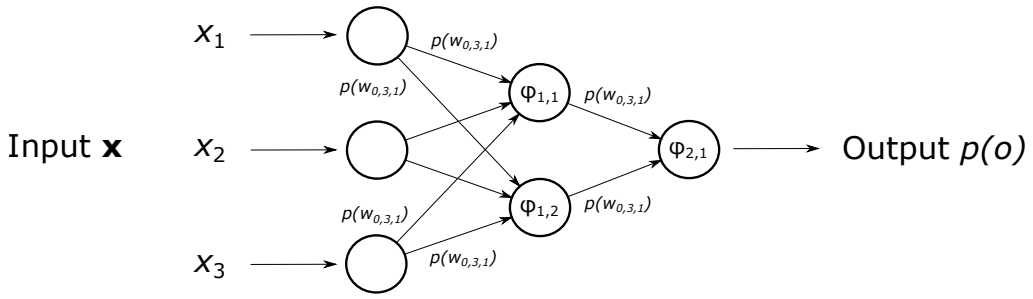


Figure 2.2: A simple probabilistic fully connected feedforward neural network. As weights w are described as a probability distribution, the model also returns a probability distribution of the output o.

2.1.3 Bayesian Neural Networks for Stochastic Settings

To obtain a neural network with strong stochastic expressiveness, the approach introduced in [DHLDVU17] is adopted. In addition to input $\mathbf{x}$, the authors suggest giving a stochastic variable z to the neural network. Furthermore, the output $\mathbf{y}$ of the neural network is disturbed by additive noise $\boldsymbol{\epsilon}$. The role of these variables is explained in the following.
Consider the case of a process which can be steered by D variables, denoted as $\mathbf{x} \in \mathbb{R}^{\mathrm{D}}$. For every input $\mathbf{x}$ we observe the outcome of the process in the form of B variables, denoted as $\mathbf{y} \in \mathbb{R}^{\mathrm{B}}$. After conducting this experiment N times and noting the pairs of inputs and outputs in matrix notation, we obtain a dataset of matrices $\mathbf{X} \in \mathbb{R}^{\mathrm{NxD}}$ and $\mathbf{Y} \in \mathbb{R}^{\mathrm{NxB}}$. Now we want to fit a neural network $f_{\mathrm{NN}} : \mathbb{R}^{\mathrm{D}} \rightarrow \mathbb{R}^{\mathrm{B}}$, parameterized by a set of weights $\mathcal{W}$, to approximate the process:

$$\mathbf{y}_n = f_{\mathrm{NN}}(\mathbf{x}_n; \mathcal{W}) \text{ for } n = 1, ..., N. \tag{2.10}$$

In practice, the process may not solely depend on the steering variables $\mathbf{x}$, but also be influenced by variables which we can neither observe nor sense. Therefore, we define a set of N random variables $\mathbf{z}$ to model the unobserved influences on the process, and assign one $z \sim \mathcal{N}(0, \gamma)$ to each of the N process observations. To account for systematic uncertainty during observations, we can use multidimensional additive output noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ with the diagonal covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{\mathrm{BxB}}$. The mapping is then described as

$$\mathbf{y}_n = f_{\mathrm{NN}}(\mathbf{x}_n, z_n; \mathcal{W}) + \boldsymbol{\epsilon} \text{ for } n = 1, ..., N. \tag{2.11}$$

To make $\boldsymbol{\epsilon}$ account for uncertainty while observing the process, its multivariate Gaussian distribution in terms of mean and covariance have to be fitted to the dataset. Furthermore, for every pair $\{x_n, y_n\}$, the mean and variance of the corresponding stochastic input z_n have to be fitted in order to obtain the unobserved feature that influenced the process during the n-th observation. Hence, the task

evolves from only fitting the weights $\mathcal{W}$ to also fitting the additive noise $\boldsymbol{\epsilon}$ globally as well as a stochastic feature z for every observation. This way, the neural network can adjust its weights so that it generates $\mathbf{y}$ from the modified input, which consists of $\mathbf{x}$ and the learned feature z. This makes it possible for the neural network to model complex stochastic effects like multimodality and heteroscedasticity [DHLDVU17, DHDVU18], as shown in Chap. 1.

Bayesian Inference and Prediction

After having collected a set of N samples $\mathbf{X}$ and corresponding labels $\mathbf{Y}$, the posterior distribution of weights $\mathcal{W}$ and stochastic features $\mathbf{z}$ is given by Bayes' rule as

$$p(\mathcal{W}, \mathbf{z} \,|\, \mathbf{X}, \mathbf{Y}) = \frac{p(\mathbf{Y} \,|\, \mathcal{W}, \mathbf{z}, \mathbf{X})\, p(\mathcal{W})\, p(\mathbf{z})}{p(\mathbf{Y} \,|\, \mathbf{X})}, \tag{2.12}$$

assuming $\mathbf{X}$ is independent of prior distributions $p(\mathcal{W})$ and $p(\mathbf{z})$, which are defined as

$$p(\mathcal{W}) = \prod_{l=0}^{L} \prod_{i=1}^{U_l} \prod_{j=1}^{U_{l+1}-1} \mathcal{N}(w_{l,ij} \,|\, 0, \lambda) \quad \text{and} \quad p(\mathbf{z}) = \prod_{n=1}^{N} \mathcal{N}(z_n \,|\, 0, \gamma). \tag{2.13}$$

The neural network has L computation-performing layers, with U_l denoting the amount of neurons in layer l inclusive of one bias-unit per layer. The expression $\mathcal{N}(x \,|\, \mu, \sigma^2)$ gives the probability that x was generated by a normal distribution with a mean of μ and a standard deviation of σ. Hence, the likelihood of $\mathbf{Y}$ is given as

$$p(\mathbf{Y} \,|\, \mathcal{W}, \mathbf{z}, \mathbf{X}) = \prod_{n=1}^{N} \mathcal{N}\big(\mathbf{y}_n \,|\, f_{\mathrm{NN}}(\mathbf{x}_n, z_n; \mathcal{W}), \boldsymbol{\Sigma}\big). \tag{2.14}$$

After having inferred the posterior probability distribution in (2.12) and the covariance matrix $\boldsymbol{\Sigma}$, the probabilities of outputs $\mathbf{y}^*$, when a new observation $\mathbf{x}^*$ is available, can in principle be calculated according to the posterior predictive distribution:

$$p(\mathbf{y}^* \,|\, \mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \\ \iiint_{\mathcal{W}\, \mathbf{z}\, z^*} \mathcal{N}\big(\mathbf{y}^* \,|\, f_{\mathrm{NN}}(\mathbf{x}^*, z^*; \mathcal{W}), \boldsymbol{\Sigma}\big)\, \mathcal{N}(z^* \,|\, 0, \gamma)\, dz^*\, p(\mathcal{W}, \mathbf{z} \,|\, \mathbf{X}, \mathbf{Y})\, d\mathbf{z}\, d\mathcal{W}. \tag{2.15}$$

Integration with respect to all possible sets of weights $\mathcal{W}$, each weighted by its probability $p(\mathcal{W}, \mathbf{z} \,|\, \mathbf{X}, \mathbf{Y})$, accounts for the uncertainty of the estimated neural network

model. Integration with respect to $\mathbf{z}$ is performed to obtain as a weighting factor the probability that the current set of weights generated $\mathbf{Y}$ for every possible configuration of the set of stochastic features $\mathbf{z}$. As the hidden feature z^* is not known for the new observation x^*, the neural network's output is integrated over all possible z^*. For each z^*, the evaluation is weighted with z^*'s probability of occurrence according to the prior distribution $\mathcal{N}(0, \gamma)$ of z, by using the factor $\mathcal{N}(z^* \,|\, 0, \gamma)$.
However, evaluating the posterior predictive distribution in such detail is not possible as computational power is limited. In addition, the posterior distribution cannot in general be calculated analytically, as the denominator in (2.12), the probability of $\mathbf{Y}$ given $\mathbf{X}$ irrespective of model parameters, easily becomes intractable. Instead, the exact posterior distribution $p(\mathcal{W}, \mathbf{z} \,|\, \mathbf{X}, \mathbf{Y})$ is approximated by a tractable distribution $q(\mathcal{W}, \mathbf{z})$, as defined in (2.17).
Having inferred q, a predictive distribution in the form of K outputs $\mathbf{y}^*$ can be obtained by executing K deterministic neural networks with weights $\mathcal{W}_k \sim q(\mathcal{W}, \mathbf{z})$ on K samples $(\mathbf{x}^*, z_k^*)$ with $z_k^* \sim \mathcal{N}(0, \gamma)$. With $\mathbf{\Sigma}$ fitted to the dataset, the output of every deterministic neural network is overlaid by noise $\boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$:

$$\mathbf{y}_k^* = f_{\mathrm{NN}}(\mathbf{x}^*, z_k^* \,; \mathcal{W}_k) + \boldsymbol{\epsilon}_k \text{ for } k = 1, ..., K. \tag{2.16}$$

Optimization

The probability distribution q, for approximating the real distribution p, is defined as a Gaussian factorization:

$$q(\mathcal{W}, \mathbf{z}) = \prod_{l=0}^{L} \prod_{i=1}^{U_l} \prod_{j=1}^{U_{l+1}-1} \mathcal{N}(w_{l,ij} \,|\, m_{w_{l,ij}}, v_{w_{l,ij}}) \prod_{n=1}^{N} \mathcal{N}(z_n \,|\, m_{z_n}, v_{z_n}). \tag{2.17}$$

To obtain the means m and the variances v of the Gaussian factors of q, black-box α-divergence minimization [HLR+16] is used to minimize a divergence between p and the approximation q. This is achieved by optimizing an energy function, whose minimizer corresponds to a local minimization of α-divergences, with one α-divergence for each of the N likelihood factors in (2.14) [DHLDVU17].
Due to the complexity of this method, the reader interested in the theory of BB-α is referred to [DHLDVU17] for additional information on applying BB-α to a neural network as used here, to [HLR+16] for the derivation of BB-α and to [Min05] for minimization of α-divergences.
The resulting energy function which has to be minimized to obtain means and variances of the factorized Gaussian distribution q is now given directly without derivation, drawing from results by [DHLDVU17] and [HLR+16].
Since q as well as the priors $p(\mathcal{W})$ and $p(\mathbf{z})$ are Gaussian distributed, there exists a proportionality

$$q(\mathcal{W}, \mathbf{z}) \propto \left[\prod_{n=1}^{N} f(\mathcal{W}) f(z_n)\right] p(\mathcal{W})\, p(\mathbf{z}), \tag{2.18}$$

with Gaussian factors $f(\mathcal{W})$ and the $f(z_n)$ that approximate the N likelihood factors in (2.14):

$$f(\mathcal{W}) = \exp\left\{\sum_{l=0}^{L}\sum_{i=1}^{U_l}\sum_{j=1}^{U_{l+1}-1}\frac{1}{N}\left(\frac{\lambda v_{w_{l,ij}}}{\lambda - v_{w_{l,ij}}} w_{l,ij}^2 + \frac{m_{w_{l,ij}}}{v_{w_{l,ij}}} w_{l,ij}\right)\right\} \propto \left[\frac{q(\mathcal{W})}{p(\mathcal{W})}\right]^{\frac{1}{N}} \tag{2.19}$$

$$f(z_n) = \exp\left\{\frac{\gamma v_{z_n}}{\gamma - v_{z_n}} z_n^2 + \frac{m_{z_n}}{v_{z_n}} z_n\right\} \propto \frac{q(z_n)}{p(z_n)}. \tag{2.20}$$

The energy function, which is minimized to adjust the Gaussian parameters of $f(\mathcal{W})$ and $f(z_n)$, is given as

$$E_\alpha(q) = -\log Z_q - \frac{1}{\alpha}\sum_{n=1}^{N} \log \mathbb{E}_{\mathcal{W}, z_n \sim q}\left[\left(\frac{\mathcal{N}\big(\mathbf{y}_n \mid f_{\mathrm{NN}}(\mathbf{x}_n, z_n; \mathcal{W}), \mathbf{\Sigma}\big)}{f(\mathcal{W}) f(z_n)}\right)^{\alpha}\right]. \tag{2.21}$$

The logarithm of Z_q is given as

$$\log Z_q = \sum_{l=0}^{L}\sum_{i=1}^{U_l}\sum_{j=1}^{U_{l+1}-1}\left[\frac{1}{2}\log\left(2\pi v_{w_{l,ij}}\right) + \frac{m_{w_{l,ij}}^2}{v_{w_{l,ij}}}\right] + \sum_{n=1}^{N}\left[\frac{1}{2}\log\left(2\pi v_{z_n}\right) + \frac{m_{z_n}^2}{v_{z_n}}\right] \tag{2.22}$$

and acts as a normalization constant of q. The minimization of (2.21) is in practice done by approximating the expectations over q with an average over K samples drawn from q. The estimated energy function is then

$$\hat{E}_\alpha(q) = -\log Z_q - \frac{1}{\alpha}\sum_{n=1}^{N} \log \frac{1}{K}\sum_{k=1}^{K}\left(\frac{\mathcal{N}\big(\mathbf{y}_n \mid f_{\mathrm{NN}}(\mathbf{x}_n, z_{n,k}; \mathcal{W}_k), \mathbf{\Sigma}\big)}{f(\mathcal{W}_k) f(z_{n,k})}\right)^{\alpha}. \tag{2.23}$$

This approach scales to large datasets as the energy function can be minimized by gradient-descent. To enable computation of gradients of $\hat{E}_\alpha(q)$, samples of the weights $\mathcal{W}$ and of $\mathbf{z}$ can be drawn from its reparameterized Gaussian parameters [KSW15]. According to this procedure, a weight $w \sim \mathcal{N}(\mu, \sigma^2)$, hence represented as $w(\mu, \sigma^2)$, is reparameterized as $w(\xi) = \mu + \sigma\xi$, with $\xi \sim \mathcal{N}(0, 1)$.
A formulation of this energy function tailored to optimization by minibatch gradient-descent [Rud16] speeds up minimization and is provided in [DHLDVU17].

2.2 Policy Search by Reinforcement Learning

The problem of policy search within the scope of this study consists of finding a way to control a system or behave in a certain environment so that some optimality criterion is maximized.
Consider a dynamical system that is described by a state $s \in \mathbb{R}$ and acted upon according to a control policy $\pi : \mathbb{R} \to \mathbb{R}$. Actions $a \in \mathbb{R}$ calculated according to this control policy enforce state transitions. As elucidated in Sec. 2.1.3 with the aid of an arbitrary process, dynamical systems too may be perturbed by unobserved disturbances $z \in \mathbb{R}$. Hence, the system may be described by a state transition function $f : \mathbb{R}^3 \to \mathbb{R}$, given as

$$s_{t+1} = f(s_t, a_t, z). \tag{2.24}$$

To have a reference as to how well different control policies perform on a system like this, a cost function $c(s_t, a_t, s_{t+1})$ is necessary, which returns a measure of the cost for every pair of action a_t and resulting state transition from s_t to s_{t+1}. The cost may depend solely on the current state transition or also take into account the sequence of states visited before or thereafter. The cumulative cost C after having the control policy π steer the system for T time steps can be calculated as

$$C(\pi) = \sum_{t=0}^{T} c(s_t, a_t, s_{t+1}). \tag{2.25}$$

The aim is to find the policy π^* that minimizes the cumulative cost:

$$\pi^* = \min_{\pi} C(\pi). \tag{2.26}$$

The control policy can be any model that maps a given input, for example the current state s_t of the system, to an action a_t.

If the circumstances allow the control agent to freely explore the system space without doing harm, then the control agent can learn the costs of different state-action pairs just by interacting with the system and eventually infer a good control policy π to act by, once most of the system space and the corresponding costs is well known. This approach is known as model-free reinforcement learning or model-free policy search and can be used for training virtual agents in computer games, as an example.
In other settings, it is not possible to explore the entire system space without doing harm. This is most often the case in real-world control settings. If a control policy were to be derived by exploring the dynamics of an in-operation district heating network, the heat demand of the consumers would probably not be met at times or production facilities might be damaged. In real-world autonomous driving, model-free reinforcement learning would very probably lead to accidents. In these cases,

the state-transition function f can be estimated from data of the real system in order to obtain a virtual model of the system. This way, control policies can be evaluated without causing harm. Hence, the control agent can interact with the virtual system and optimize the policy to minimize a chosen cost function. This procedure is known as model-based reinforcement learning or model-based policy search. The upcoming chapter elucidates a model-based policy search algorithm which deploys a Bayesian neural network with stochastic inputs, see Sec. 2.1.3, as model to optimize a policy π which is parameterized as a deterministic neural network, see Sec. 2.1.1.

Chapter 3

Technical Approach

The model-based policy search method detailed in this chapter can in principle be applied to any steerable dynamical system in which at least part of the state variables can be observed. This chapter is divided into two parts. Building upon the theoretical methods introduced in Chap. 2, the first part gives the generic control approach while the second part details the specific implementation tailored to the district heating system.

3.1 Bayesian Neural Network Based Policy Search

This section starts with introducing suitable terminology for defining a class of steerable and partially observable dynamical systems. Subsequently, it is detailed how the Bayesian neural network is trained to approximate the dynamical system. Eventually, the policy search itself is elucidated. The section closes by summarizing the entire procedure in an algorithmic formulation.

Dynamical System

As in Sec. 2.2, consider a discrete dynamical system that is described by a state vector $\mathbf{s}_t \in \mathbb{R}^{\mathrm{B}}$. At time t, the system can be acted upon using actions $\mathbf{a}_t \in \mathbb{R}^{\mathrm{D}}$. The dynamics f of the system is corrupted by a disturbance z. The class of dynamical systems dealt with in this study suffer from time delays. That means that the state $\mathbf{s}_t$ and action $\mathbf{a}_t$ do not suffice but a sequence of previous states and actions is necessary to infer the next state $\mathbf{s}_{t+1}$. For the sake of a clear notation, a concatenated vector $\mathbf{x}_t = [\mathbf{s}_t, \mathbf{a}_t] \in \mathbb{R}^{\mathrm{BD}}$ is introduced and henceforth referred to as process vector. The sequence $\mathbf{X}_P \in \mathbb{R}^{\mathrm{PxBD}}$ of process vectors of previous time steps $[t-P, ..., t]$ is accordingly referred to as the process sequence. With this terminology, a dynamical system perturbed by a disturbance z and time delays can be mathematically defined as

$$\mathbf{s}_{t+1} = f(\mathbf{X}_P, z), \tag{3.1}$$

where the sequence length P is the maximum time delay in a specific system.

Model Training

A model that approximates the real system dynamics is required for model-based policy search. A reasonable approach to obtain a model is to collect a dataset of the real system dynamics, pairs of samples $\mathbf{X}_P$ and corresponding labels $\mathbf{s}_{t+1}$, that is, and subsequently make the model approximate the dynamics f by applying supervised machine learning. However, rolling out the system dynamics requires some existing control policy to steer the system by choosing actions $\mathbf{a}_t$. Yet, a simple controller that keeps the system variables in the admissible range can be used to obtain a dataset that represents part of the system dynamics space. When applying the control policy, the resulting system evolution can be logged and ultimately recast to the sample-label notation, as detailed in Sec. 2.1.3, for training the model.
The model is parameterized as a Bayesian neural network with feature and observation noise as introduced in Chap. 2. This choice is motivated by two reasons. Neural networks in general are shown capable of accurately approximating a wide range of mathematical functions. Complementarily, Bayesian inference and the random variables z which are given as additional input to the neural network in combination with the additive output noise $\boldsymbol{\epsilon}$ are expected to mitigate effects of observation and feature uncertainty, as elucidated in Sec. 2.1.3.
For the proposed Bayesian neural network, the parameters to be optimized amount to means and variances of the weights $\mathcal{W}$, means and variances of the set of random inputs $\mathbf{z}$, and the covariance matrix of the additive output noise $\boldsymbol{\epsilon}$. These parameters are optimized by Bayesian inference as detailed in Sec. 2.1.3.

Policy Search

The trained model is used to simulate the system dynamics and thereby assess what sequence of system states is likely to arise for an arbitrary control policy π. Predictions can be made with the Bayesian neural network by using the posterior predictive distribution (2.15). With the learned distribution q over weights and random inputs, probabilities of ensuing system states $\mathbf{s}_{t+1}$ can be calculated as

$$p(\mathbf{s}_{t+1} \,|\, \mathbf{X}_P) = \iiint_{\mathcal{W}\ \mathbf{z}\ z^*} \mathcal{N}\big(\mathbf{s}_{t+1} \,|\, f_{\mathrm{NN}}(\mathbf{X}_P, z^*; \mathcal{W}), \boldsymbol{\Sigma}\big)\, \mathcal{N}(z^* \,|\, 0, \gamma)\, dz^*\, q(\mathcal{W}, \mathbf{z})\, d\mathbf{z}\, d\mathcal{W}. \tag{3.2}$$

Generating system rollouts involves heavy use of the model, as for every time step a distribution of the next system state $\mathbf{s}_{t+1}$ has to be calculated. As this is com-

putationally expensive, the approach by [DHLDVU17] is followed and (3.2) is approximated by sampling K sets of parameters of q, as described in the derivation of (2.16) in Sec. 2.1.3. Hence, K outputs $\mathbf{s}_{t+1,k}$ as

$$\mathbf{s}_{t+1,k} = f_{\mathrm{NN}}(\mathbf{X}_P, z_k^*\,; \mathcal{W}_k) + \boldsymbol{\epsilon}_k \text{ for } k = 1, ..., K, \tag{3.3}$$

with $z_k^* \sim \mathcal{N}(0, \gamma)$, are calculated to constitute an approximate predictive distribution of the next state.
The control policy π is a deterministic neural network, as introduced in Sec. 2.1.1, thus the policy is parameterized by a set of scalar weights $\mathcal{W}_\pi$. The control policy computes an action $\mathbf{a}_{t+1}$ given previous actions and states, which are for the sake of notation again wrapped up in the process sequence $\mathbf{X}_P$, as

$$\mathbf{a}_{t+1} = \pi(\mathbf{X}_P, \mathbf{s}_{t+1}; \mathcal{W}_\pi). \tag{3.4}$$

Having the statistical model and policy interact alternatingly generates K system rollouts. If the policy is rolled out over a future horizon of H time steps, then each rollout yields a process sequence $\mathbf{X}_{P+H}$. The control policy is then optimized by reinforcement learning, as detailed in Sec. 2.2. Therefore, a cost function $c(\mathbf{x}_t)$ that assigns a cost to each process vector $\mathbf{x}_t$ is required. The cost function should be in congruence with the desired control aims and potential constraints of the system. For K system rollouts over a future horizon H, the average cost per state is calculated as

$$C(\pi) = \frac{1}{KH} \sum_{k=0}^{K} \sum_{t=0}^{H} c(\mathbf{x}_{t,k}^{\pi}), \tag{3.5}$$

with $\mathbf{x}_t^\pi$ being process vectors obtained by applying policy π, that is, actions $\mathbf{a}_t$ calculated with policy π and states $\mathbf{s}_t$ that are computed in response by the model. The objective for the policy, as given in (2.26), is minimizing the cost. As both the Bayesian neural network as model and the deterministic neural network as policy are differentiable, gradients of the cost with respect to the weights $\mathcal{W}_\pi$ of the policy's neural network can be calculated and used to update these weights in a cost-decreasing manner, as briefly addressed in Sec. 2.1.1.
For deriving a useful control policy with this procedure, it is essential that the model accurately simulate the real system dynamics. However, the procedure as introduced so far relies on applying an initial control policy to the real system to generate a dataset for training the model. As it cannot be generally expected that the initial control policy is very sophisticated, the controller might only choose actions from a rather limited interval. Therefore, parts of the action-space and the state-space are not covered in the resulting dataset, and the learned dynamics model may not generalize well. Yet, it may be expected that a more sophisticated control policy will be obtained during model-based policy search. In case a wider range of possible actions and states is explored by the updated control policy, the problem of limited

Algorithm 1 Bayesian neural network based policy search

1: Obtain a dataset $\{\mathbf{X}_P^n, \mathbf{s}_{t+1}^n\}_{n=0}^N$ by applying a control policy to the system f
2: Minimize (2.23) to fit $q(\mathcal{W}, \mathbf{z})$ and $\mathbf{\Sigma}$
3: Sample sets of weights $\{\mathcal{W}_k\}_{k=0}^K$ from $q(\mathcal{W}, \mathbf{z})$
4: Initialize policy weights $\mathcal{W}_\pi$
5: **for** $i = 1 : I$ **do**
6: $\quad C(\pi) \leftarrow 0$
7: $\quad$ **for** $n = 1 : N$ **do**
8: $\quad\quad$ **for** $k = 1 : K$ **do**
9: $\quad\quad\quad$ Retrieve $\mathbf{X}_P^n$ from the dataset
10: $\quad\quad\quad$ **for** $h = 1 : H$ **do**
11: $\quad\quad\quad\quad$ Sample z from $\mathcal{N}(0, \gamma)$
12: $\quad\quad\quad\quad$ Sample $\boldsymbol{\epsilon}$ from $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$
13: $\quad\quad\quad\quad$ $\mathbf{s}_{t+1}^n \leftarrow f_{\mathrm{NN}}(\mathbf{X}_P^n, z\,; \mathcal{W}_k) + \boldsymbol{\epsilon}$
14: $\quad\quad\quad\quad$ $\mathbf{a}_{t+1}^n \leftarrow \pi(\mathbf{X}_P^n, \mathbf{s}_{t+1}^n; \mathcal{W}_\pi)$
15: $\quad\quad\quad\quad$ $\mathbf{x}_{t+1}^n \leftarrow [\mathbf{s}_{t+1}^n, \mathbf{a}_{t+1}^n]$
16: $\quad\quad\quad\quad$ $\mathbf{X}_P^n \leftarrow [\mathbf{X}_{P-1}^n, \mathbf{x}_{t+1}^n]$
17: $\quad\quad\quad\quad$ $C(\pi) \leftarrow C(\pi) + c(\mathbf{x}_{t+1}^n)$
18: $\quad C(\pi) \leftarrow \frac{C(\pi)}{KH}$
19: $\quad$ Update $\mathcal{W}_\pi$ in direction of $-\frac{dC(\pi)}{d\mathcal{W}_\pi}$
20: Stop or repeat from line 1 with obtained control policy $\pi(\,\cdot\,; \mathcal{W}_\pi)$

state and action space exploration may be mitigated by applying this control policy to the real system to generate a new dataset. With the new dataset, a more general and accurate model may be obtained and this enables deriving a control policy that is more accurate and leverages a bigger state and action space of the dynamic system.
The entire procedure of model-based policy search as detailed in this section is given step by step in Alg. 1.

3.2 Implementation

The introduced approach is applied to derive a control policy for a district heating network. This section details the implementation specific to the benchmark district heating network that is introduced in Sec. 1.2. After giving the actions and the states of the district heating network, the control policies used to generate a training dataset are detailed. Subsequently, implementation and training of the Bayesian neural network as model and the deterministic neural network as control policy are detailed.

States	**s**	**Actions**	**a**
Supply water temperature	T_{sup}	Thermal power	Q
Tank water temperature	T_{tank}	Control valve	v
Return water temperature	T_{ret}		

(a) Overview of the states **s** and actions **a** that describe the district heating system.

	Model	**Control Policy**
Parameterization	Bayesian neural network	Deterministic neural network
Hidden activation	ReLU [NH10]	ReLU or tanh
Output activation	None	None
Sample shape	$[\mathbf{X}_P, \tau, T_{\text{ext}}, d]$	$[\mathbf{X}_P, \mathbf{s}^*_{t+1}, \tau, T_{\text{ext}}, d]$
Output	$\mathbf{s}^*_{t+1}$	$\mathbf{a}_{t+1}$
Input standardization	Gaussian [KKN11]	Gaussian
Output standardization	Gaussian	Logistic
Optimization loss	BB-α Energy (2.23)	$C(\pi)$ (3.5)
Gradient calculation	Autograd [MDA15]	Autograd
Parameter updates	Adam [KB14]	Adam

(b) Implementation and optimization of model and control policy

Table 3.1: Information about the implementation of the system, the model, and the control policy. Hidden and output activation denotes the activation functions of neurons in hidden and output layers, respectively. The sample is the data vector given as input to the neural networks. Next to the process sequence $\mathbf{X}_P$, the samples incorporate daytime τ [min], external temperature T_{ext} and the day of the week d.

Table 3.1a provides an overview of the system description. The control input to the system can be updated every 15 minutes. The system's state vector **s** comprises the temperatures of the water that is supplied by the production unit T_{sup}, the water stored in the tank T_{tank} and the water that is returned to the production unit after traversing the distribution network, T_{ret}. The action vector **a** contains the amount of thermal power production Q to heat up water and the position of the valve v that regulates the water flow through the heat tank.

3.2.1 Training Dataset Generation

Different control policies are deployed to generate data of the system dynamics for training the Bayesian neural network model. The character of the dataset has a strong influence on the model performance, for it cannot be expected that the model also generalizes to dependencies and phenomena which are not present in the training dataset. To obtain an expressive dataset — one that explores a wide range

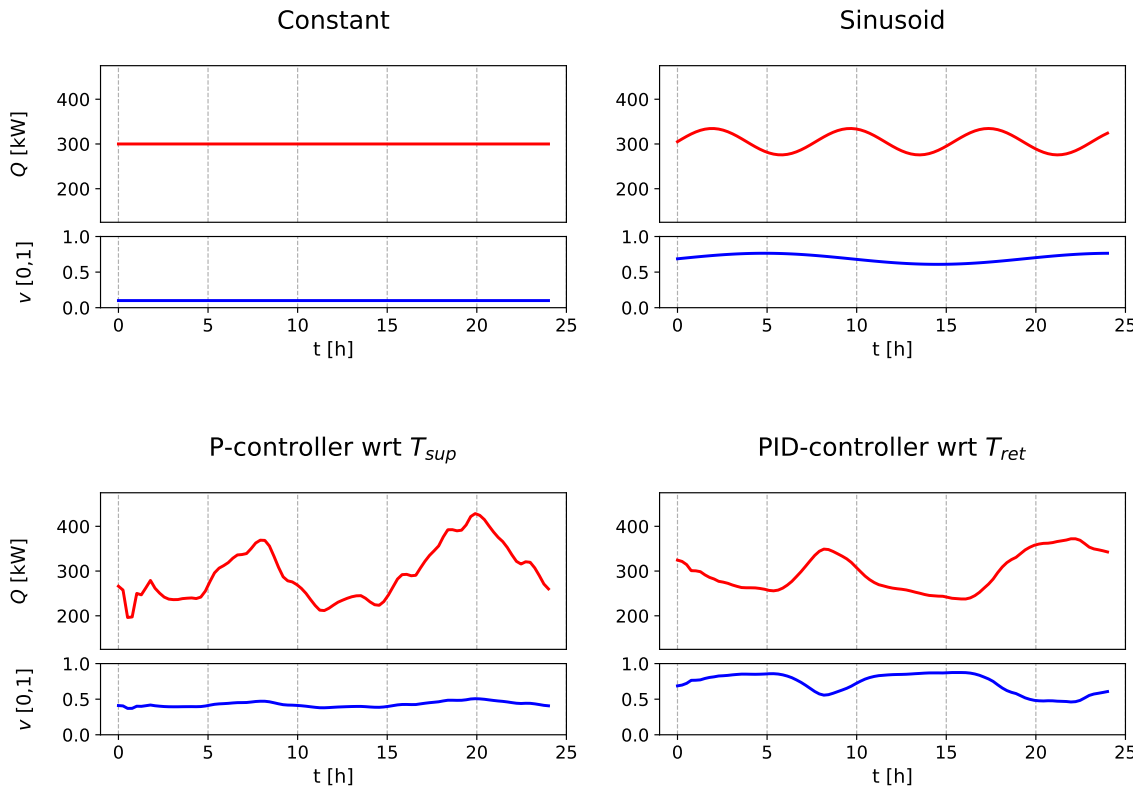


Figure 3.1: Initial control policies used to generate a dataset for training the Bayesian neural network. Exemplary actions Q and v computed by four different control policies are depicted.

of the admissible state and action space, and contains the system's stochasticity as well as the dependencies between actions and states —, four different control policies are deployed. While every control policy follows a certain scheme, their exact parameters are updated every one or two weeks of the simulation time. Exemplary actions $\mathbf{a}$ specified by the control policies are sketched in Fig. 3.1.
As the name implies, the constant control policy gives constant control inputs. A parameter update of this control policy comprises shifting both the thermal power Q and the valve position v. The sinusoid control policy generates sinusoidal control inputs. Parameter updates comprise resampling baselines, amplitudes and modifying the periods of both control inputs' sinus curves. The proportional (P)-controller tries to keep the temperature of the supplied water T_{sup} close to a specified control aim $T_{\text{sup,aim}}$:

$$\mathbf{a}_t = \mathbf{k}_{\text{p}}\big(T_{\text{sup}} - T_{\text{sup,aim}}\big). \tag{3.6}$$

Updating the parameterization here accounts for modifying the value of the control aim $T_{\text{sup,aim}}$. The fourth control policy is a proportional integral derivative (PID)-controller with respect to the temperature of the water returned to the production

site T_{ret}:

$$\mathbf{a}_t = \mathbf{k}_\text{p} E_t + \mathbf{k}_\text{i} \int_{-\infty}^{t} E_t \, dt + \mathbf{k}_\text{d} \frac{dE_t}{dt} \quad \text{with } E_t = T_\text{ret} - T_\text{ret,aim} \tag{3.7}$$

Updating its parameterization comprises modifying the control aim $T_{\text{ret,aim}}$ and resampling the coefficients $\mathbf{k}_\text{p}$, $\mathbf{k}_\text{i}$ and $\mathbf{k}_\text{d}$ of the PID-controller from specified normal distributions. The means of the parameters of the P- and PID-controller are tuned by hand to obtain a trade-off between smooth control actions — that is, low values of dQ and dv in (3.8) — and convergence to the specified control aim $T_{\text{sup,aim}}$ or $T_{\text{ret,aim}}$, respectively.
The constant control policy is deployed to obtain data samples which show the development of the system states $\mathbf{s}$ independent of varying control inputs $\mathbf{a}$. This way, it may also be possible for the model to learn purely stochastic influences that do not depend on the control inputs, like the variation in the consumption curves. The sinusoid control policy is deployed to show the impact of rising and falling control inputs at arbitrary time steps on the system state. These dependencies may not be fully apparent in a dataset generated by only the P- and PID-controller, which tend to compute control inputs that mostly peak during morning and evening hours while computing low inputs during all other times of the day. These, however, are deployed to directly highlight the impact of the control inputs $\mathbf{a}$ on the supply temperature T_{sup} and the return temperature T_{ret}, respectively.

3.2.2 Model Parameterization

A Bayesian neural network, as introduced in 2.1.3, is deployed to model the system dynamics. Information about the implementation and training of the Bayesian neural network is provided in Subtab. 3.1b. To keep computational efforts tractable, the hidden neurons activate as ReLUs, whereas the output neurons do not activate and just propagate their input sum. The prior mean λ of the weights in (2.13) is initialized as 1, and the prior mean γ of the stochastic input z is initialized as the square root of the length of process vector $\mathbf{x}$, following [DHLDVU17]. One training data sample comprises the previous P states and actions, bound together in the process vector $\mathbf{X}_P$, the daytime τ of the sample, the external temperature T_{ext} and the day of the week d. Following [DHLDVU17], the model is fitted to return as output the incremental state vector $d\mathbf{s}_t = \mathbf{s}_{t+1} - \mathbf{s}_t$. Consequently, the resulting state during policy search is calculated as $\mathbf{s}_{t+1} = \mathbf{s}_t + d\mathbf{s}_t$. Both the input and the output data is standardized assuming a Gaussian distribution [KKN11], that is, subtracting mean and dividing by the respective standard deviation.
The gradients of the energy function, calculated with the automatic differentiation tool Autograd [MDA15], are used to update the parameters with the Adam optimization algorithm [KB14]. In line with the definition of the energy function (2.23),

training is performed by batch gradient-descent.

3.2.3 Control Policy Parameterization

As listed in Tab. 3.1b, the control policy is parameterized as a deterministic feed-forward fully connected neural network. Hidden neurons activate either as ReLUs or tanh-functions — results for configurations with either activation function are provided in Chap. 4. The output neurons return the sum of their input without applying an activation function. Samples given as input to the control policy comprise a process sequence $\mathbf{X}_P$, the state vector $\mathbf{s}_{t+1}$ computed by the model, and current daytime τ, external temperature T_{ext} and the day of the week d. The input is standardized according to a Gaussian distribution. Both variables in the action vector $\mathbf{a}_{t+1}$ that are returned as output by the control policy are transformed with a logistic function. The position of the control valve is thereby mapped to a value within the interval $[0, 1]$. The amount of thermal power Q is obtained by multiplying the logistic value with a defined maximum permissible value for Q.

3.2.4 Cost Function

The policy is optimized by minimizing a cost function that is in line with the control aims. The cumulative cost $C(\pi)$ (3.5) of a control policy π is defined as the average cost of the process vectors $\mathbf{x}^\pi$ that ensue by applying the policy. The cost function for process vectors is designed under the following assumptions. Firstly, the amount of produced thermal power Q should be kept within an admissible interval $[Q^-, Q^+]$. This interval depicts the boundaries that the values of Q rarely surpass for weekday-samples in the training dataset for the Bayesian neural network. The forecasts for samples that contain actions Q beyond this interval are found to be inaccurate. However, as the consumptions may drop significantly during weekends, as detailed in Sec. 1.2.2, the required thermal power Q to keep the return temperature T_{ret} close to $T_{\text{ret,aim}}$ may fall below Q^-. Therefore, this part of the cost is only used in the early stages of optimization and omitted once the policy's actions Q are on average significantly higher than Q^-. The second design guideline is that the temperature of the returned water T_{ret} should be kept constant. Ultimately, strong deviations from previous control inputs are penalized to account for physical constraints. For the sake of clarity, the variables $dQ_t^- := \max(0,\, Q^- - Q_t)$, $dQ_t^+ := \max(0,\, Q_t - Q^+)$, $dQ_t := |Q_t - Q_{t-1}|$, $dv_t := |v_t - v_{t-1}|$ and $dT_{\text{ret,t}} := |T_{\text{ret, aim}} - T_{\text{ret,t}}|$ are introduced. The cost is subdivided into the parts

$$
\begin{aligned}
c_Q &= \alpha_0 dQ_t^- + \alpha_1 dQ_t^{-2} + \beta_0 dQ_t^+ + \beta_1 dQ_t^{+2} \\
c_{dQ} &= \gamma_0 dQ_t + \gamma_1 dQ_t^2 \\
c_{dv} &= \delta_0 dv_t + \delta_1 dv_t^2 \\
c_{dT_{\mathrm{ret}}} &= \eta_0 dT_{\mathrm{ret,t}} + \eta_1 dT_{\mathrm{ret,t}}^2
\end{aligned}
\tag{3.8}
$$

that are summed up to obtain the entire cost of a process vector $\mathbf{x}_t^\pi$ generated by control policy π as

$$
c(\mathbf{x}_t^\pi) = c_Q + c_{dQ} + c_{dv} + c_{dT_{\mathrm{ret}}}. \tag{3.9}
$$

The coefficients $\alpha_0, ..., \eta_1$ used to weight the partial costs to a similar scale are optimized by fitting pairs of exemplary process vectors and corresponding desired values for the cost. The optimized coefficients are shown in Fig. 3.2.
To prefer simple over very tightly fitted neural networks, a L2-regularization term is added to the cumulative cost (3.5) [Ng04].

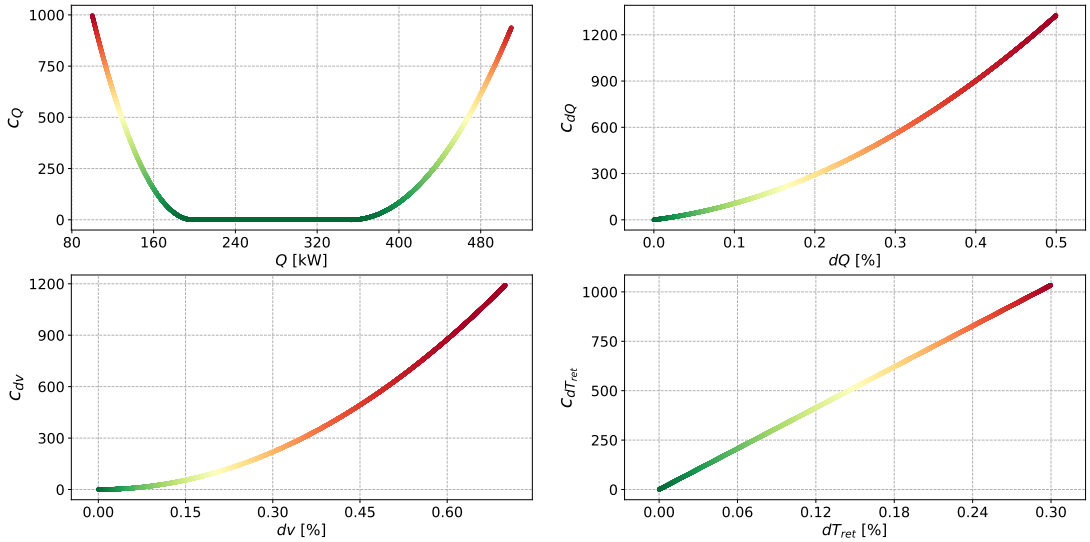


Figure 3.2: Decomposition of the cost assigned to a process vector $\mathbf{x}_t^\pi$ after optimization of the cost coefficients $\alpha_0, ..., \eta_1$ in (3.8).

3.2.5 Policy Optimization

The control policy is optimized according to Alg. 1. Batch-gradients are calculated with Autograd and Adam is used to calculate updates for the set of weights $\mathcal{W}_\pi$ in line 19 of Alg. 1. As mentioned in Sec. 1.2.2, the customers attached to the district

heating network exhibit consumption characteristics that vary according to the day of the week. To ensure that the control policy is exposed to the entire stochastic behavior of the dynamical system, process sequences $\mathbf{X}_P$, from which rollouts are generated, include all days of the week. During policy search over a horizon of length H, the daytime τ and the day of the week d are incremented while the external temperature T_{ext} is assumed to be constant.
The weights $w \in \mathcal{W}_\pi$ of the deterministic neural network, which acts as control policy, are initialized as uniform random values close to zero. As given in line 14 of Alg. 1, the action vector $\mathbf{a}_{t+1}$ is computed by the control policy π. However, for $h = 1$, the previous actions, saved in the process sequence $\mathbf{X}_P$, are computed by one of the control policies that are used to generate the training dataset. This change in control policies may lead to a significant discontinuity in the control inputs $\mathbf{a}$, and thereby in the immediate process sequence $\mathbf{X}_{P+h}$. This in turn may generate inaccurate forecasts $\mathbf{s}_{t+1}$ by the model, in line 13 for $h > 1$, as such discontinuities are not present in the training dataset. Therefore, to ensure a smooth transition between control policies, the control policy π is pre-trained to return, as action $\mathbf{a}_{t+1}$, values that are close to the previous action $\mathbf{a}_t$, before starting the policy search.
The number K of rollouts, as in line 8, is initialized with a small value and increased progressively with training iterations i. This is done to speed up the calculations of the weight updates during early stages of the policy search. At first, the control policy tends to be assigned relatively high cumulative cost values $C(\pi)$, as the generated rollouts are often far from optimal with respect to the cost function. As the cost is relatively high, meaningful gradients, i.e., the approximate direction in which the weights $\mathcal{W}_\pi$ have to be updated, can also be determined with a small K. With increasing training iterations i, the cost decreases and the parameter updates decrease. Therefore, a higher number K of rollouts is required so that the gradient of the averaged cost (3.5) still points in the correct direction despite the stochasticity of the system.

Adam parameterization In all optimizations, which are model training, policy pre-training as discussed in the previous paragraph, cost coefficient optimization, and model-based policy search, the following parameters are used for Adam:
$\alpha = 0.0001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ [KB14].

Chapter 4

Evaluation

In this chapter, the control approach is applied to the benchmark district heating network described in Sec. 1.2. A Bayesian neural network with stochastic inputs, as introduced in Sec. 2.1.3, is trained to model the district heating network dynamics. A deterministic neural network, as introduced in Sec. 2.1.1, constitutes the control policy and is optimized by model-based policy search according to Sec. 3.1.
First, results of the Bayesian neural network, the policy optimization process, and the obtained control policy are shown. The results are described and background information is provided. Subsequently, the meanings of the results are discussed and evaluated in the discussion section.

4.1 Results

Initially, the performance of the Bayesian neural network is evaluated. After examining the accuracy of modelling the stochastic consumption curves, the generation of the dataset used to train the Bayesian neural network to model the district heating system is elucidated. Then, the accuracy of the obtained model is evaluated. Subsequently, the policy optimization process based on the learned model is shown. Finally, graphs that depict the performance of the obtained control policy are provided. District heating system rollouts for both the derived control policy and the PID-controller are provided to make qualitative comparisons.

4.1.1 Learned Consumer Models

The most part of the stochasticity in the district heating network is caused by the stochastic consumer behavior, as elucidated in Sec. 1.2.2. Therefore, an experiment is conducted to evaluate whether the Bayesian neural network is able to model isolated consumption curves. In this setting, a dataset consists of three consumption curves by the same consumer for the same day of the week. The Bayesian neural network has to learn to map daytime values to corresponding consumption values.

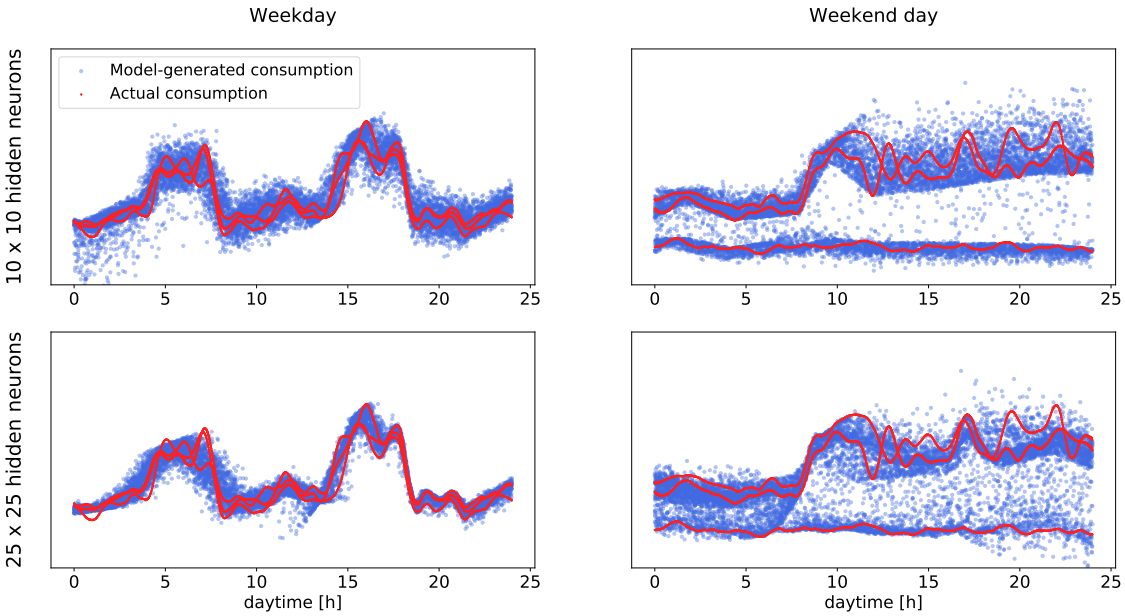


Figure 4.1: Bayesian neural networks with stochastic inputs modelling the consumption curves. This figure depicts the fit of two differently configured Bayesian neural networks on consumption curves of a weekday and a weekend day.

The results are shown in Fig. 4.1. A separate network is trained for every plot. All networks are trained by applying Adam for 10,000 epochs. The two left plots feature consumptions for a weekday, whereas the two plots on the right feature consumptions for a weekend day. The upper plots show the fit of a Bayesian neural network with 10x10 hidden neurons while the bottom plots show the fit of a network with 25x25 hidden neurons. For weekdays, the fit of the bottom network is in general tighter and generates less outliers than the top network. For the weekend day, the bimodality is more accurately captured by the top neural network than by the bottom neural network. The bottom network generates a considerable amount of outliers between the two modes, especially during the hours 0 and 7 am, where the two modes are relatively close to each other. The top network approximates the frequent variations of the sinusoid curves mostly with rectangular-like equal-density error bands. This is particularly noticeable at the lower mode over the entire time axis and the upper mode during the hours of 12 and 24. As opposed to the top network, the bottom network fits the coinciding peaks of the upper mode at 17 hours to some extent.

4.1.2 Learned Dynamics of the District Heating System

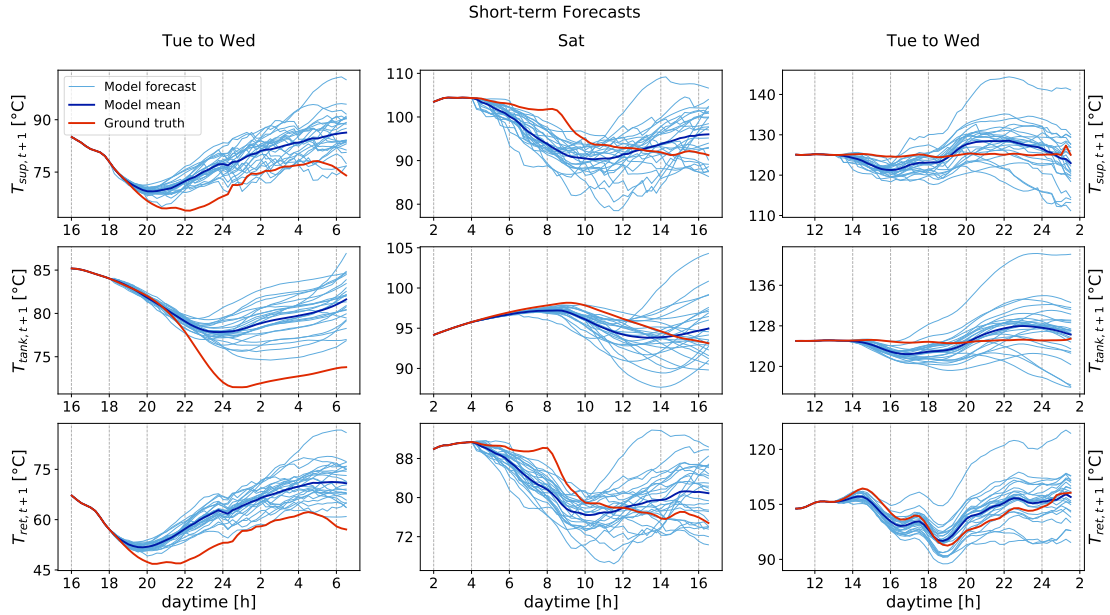
The Bayesian neural network is trained to learn the dynamics of the district heating system. The dataset generation and the performance of the obtained model is detailed.

Dataset and Training

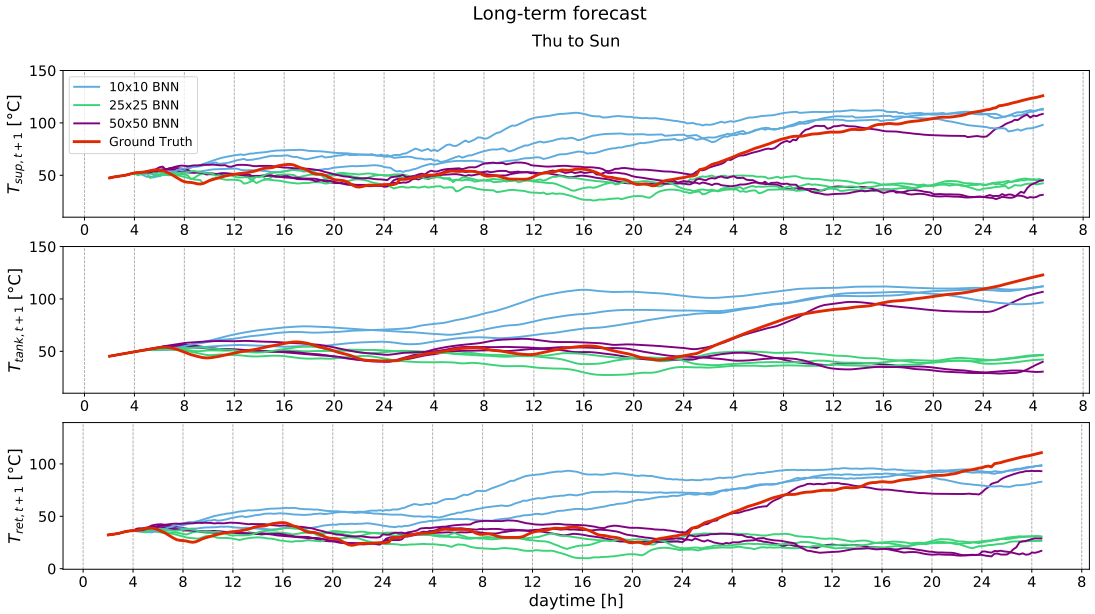
The dataset for training the Bayesian neural network is generated by simulating the benchmark district heating system for a period of 1600 days. Each of the four control policies detailed in Sec. 3.2.1 is deployed for an equal share of the entire simulation time. Sampling the system state every 15 minutes, a dataset of approximately 150,000 samples is generated, of which, after keeping a share for validation and filtering outliers of the remainder, around 135,000 samples are used for training. Outliers are here defined as samples in which the network temperatures surpass or fall below specified thresholds. Physically unrealistic temperatures ensue, for example, when the constant control policy specifies a low amount of thermal power Q during times of peak demand, or, on the other hand, specifies a high amount of thermal power on a weekend day on which by chance many consumers at once exhibit low consumptions due to bimodality.
The total time it takes until a given control input Q influences the return temperature T_{ret} is 70 minutes, as detailed in Sec. 1.2.2. A systemic validation gives that the impact of a change in control inputs Q becomes negligible after approximately 120 minutes. Therefore, the length P of the sequence of previous states is set to 9. With the sampling interval of 15 minutes, this accounts for a sequence of states and actions of the previous 135 minutes in the process sequence $\mathbf{X}_P$.
The model is trained as detailed in Sec. 3.2. The parameter K in the energy function (2.23) is set to 25. All networks are trained by applying Adam, parameterized as given in Sec. 3.2.5, for 20,000 epochs.

Accuracy of the Learned Model

To evaluate how well the model approximates the district heating system, an identical sequence of control vectors $\mathbf{a}$ is given to both the system and the model. Consequently, the sequence of state vectors $\mathbf{s}$ generated by the actual system is compared to the sequences generated by the model. Figure 4.2 shows several comparisons of the actual and the model-generated system development. Subfigure 4.2a depicts system rollouts over a half-day horizon, for three different sequences of control vectors $\mathbf{a}$. Every variable in the three-fold state vector $\mathbf{s}$ is graphed separately. Plots in the same column belong together. The days that are covered by a depicted sequence of state vectors $\mathbf{s}$ are noted on the top of its column. The daytimes are given on the horizontal axis. One plot features as red curve the actual development of the state variable as well as $K = 25$ model-generated forecasts, with the dark blue curve being the mean of all forecasts. Subfigure 4.2b compares the sequence of the actual state with the forecasts by three differently parameterized models over a horizon of three days.



(a) Short-horizon rollouts of a learned model compared to the actual system dynamics.



(b) Long-horizon rollouts of three differently parameterized models.

Figure 4.2: This figure details approximations of the system development made by learned models. Subfigure (a) shows $K = 25$ system forecasts over a horizon $H = 50$ by a Bayesian neural network with 10x10 hidden units for three different samples. Subfigure (b) shows $K = 3$ forecasts over a horizon $H = 300$ generated by three models with different hidden layer sizes for the same sample.

The left column in Subfig. 4.2a shows model-rollouts that begin at 6 pm and end at around 6 am in the following day. While the model in general reproduces the shape of the curve, the predictions of all state variables suffer from offsets that are developed between 7 pm and midnight. Around 5 am, T_{sup} and T_{ret} diverge further from the model-predicted mean. The central column depicts rollouts that start at 4 am on Saturday. While the exact shape of the real curve is missed by the forecasts, the trend is followed on average. The right column shows rollouts generated between noon and midnight of a weekday. The sequence of actions is specified by the P-controller. For T_{sup} and T_{tank}, the model generates curves that follow the trend of T_{ret} even though the P-controller keeps these rather constant. In the district heating system, the trend of T_{sup} is strongly influenced by the trend of T_{ret} for the other control policies that are deployed to generate the dataset, as seen in the two left columns. However, this does not hold for the P-controller that tries to keep T_{sup} close to $T_{\mathrm{sup,aim}}$. The trend of the return temperature T_{ret}, on the other hand, is closely approximated by the model mean. Apart from the control inputs, the driving force of the system is the aggregate consumption of the attached entities. Despite the incorporated stochasticity, individual consumptions show distinct maxima around 9 am and 7 pm, as seen in Fig. 1.5a. Correspondingly, the visible minimum of the return temperature around 7 pm reflects the usual course of T_{ret} on weekdays, due to the preceding afternoon consumptions. This minimum is equally visible in the rollouts in the left column.
Subfigure 4.2b shows $K = 3$ forecasts of the system development ranging from Thursday 4 am until Sunday 4 am generated by three Bayesian neural networks with 10x10, 25x25 and 50x50 hidden units. Until the night from Friday to Saturday, both the models with 25x25 and 50x50 hidden units in general follow the real curve, while the rollouts generated by the 10x10 model drift away. After that, the real curve steadily rises to eventually cross the rollouts generated by the 10x10 model. Apart from one rollout of the 50x50 model, which follows the tendency of the real curve also after the turning point, the remaining rollouts, as well as the rollouts by the 25x25 model, fail to follow. The control inputs Q and v used to generate these rollouts are constant. Recurrent lows of T_{ret} in the morning and the evening, as given by the real curve due to aggregate consumption peaks, are temporarily apparent in the rollouts generated by the 50x50 model, and roughly visible in the rollouts by the 10x10 model, yet stretched in time.

Rollouts generated by the Bayesian neural network with 50x50 hidden units for three different sequences of actions $\mathbf{a}$ that cover weekend days are provided in Fig. 4.3. For the sample in the left column, the forecasts by the model split up into two directions at around midnight. This is congruent with the bimodal consumption curves in Fig. 1.5b. In the central column, the forecast mean accurately follows the real curve until 8 am. After that, the individual forecasts diverge while the mean of all forecasts continues to follow the tendency of the real curve. In the right column, the forecasts initially exert low a variance initially and are split up into two groups

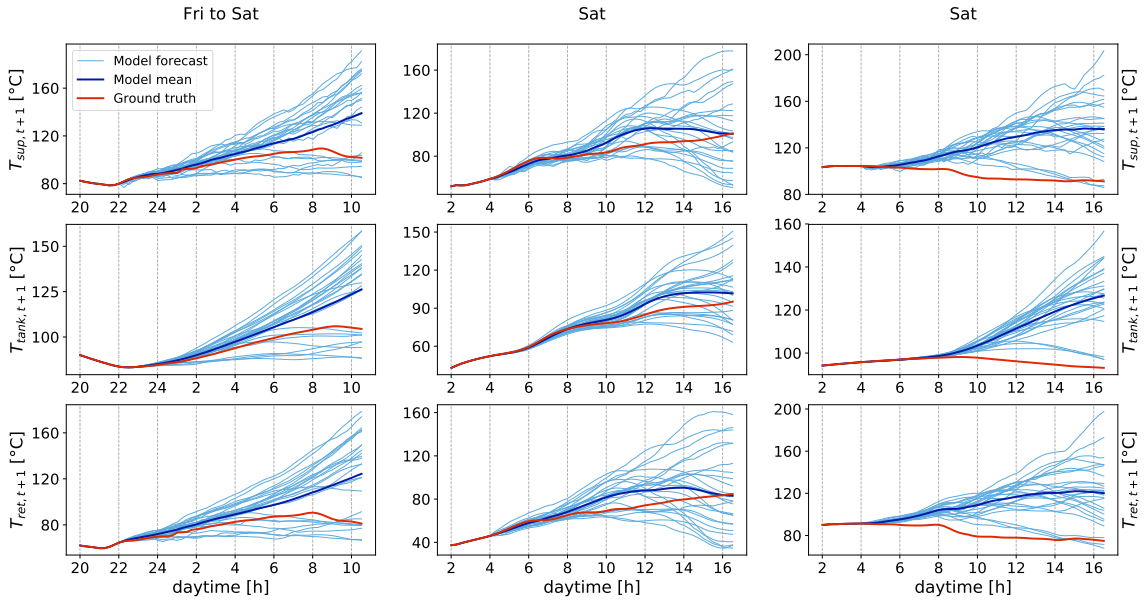


Figure 4.3: Rollouts that cover weekend days generated by a Bayesian neural network with 50x50 hidden units.

between 9 and 12 am, where the exact timing depends on the state variable. While a small share of the rollouts follows the trend of the real curve, the majority drifts away into higher temperatures. This phenomenon is also present in the replication of the bimodal consumption curves in Fig. 4.1 by the Bayesian neural network with 25x25 hidden units. The 25x25 model barely distinguishes the two modes between midnight and 7 am. With the rise of the top mode at 8 am, a connection between the lower and the upper mode is established in the model. The timing is similar to the bifurcation point shown by the 50x50 model in the right column of Fig. 4.3. However, this is not congruent with the real consumer behavior, as the consumption curves are designed such that it is already apparent at midnight which mode is entered, as graphed in Fig. 1.5b. For a comparison, the same sequence of actions $\mathbf{a}$ is also used in the central column of Subfig. 4.2a that depicts rollouts by the 10x10 model. In the replication of the 10x10 model in Fig. 4.1, the two modes are clearly separated. Consequently, a bifurcation point at 8 am cannot be observed in the central column of Subfig. 4.2a.

In the left and the right column of Fig. 4.3, the variation of the rollouts in the lower mode is rather constant while the variation in the upper mode increases with time. This is congruent with the heteroscedasticity in the consumption curves of weekend days. As seen in Fig. 1.5b, the upper mode exhibits a stronger variation than the lower mode.

Activation	Hidden units	MSE-S	MSE-L
ReLU	30	244 ± 26	$7\text{x}10^{12} \pm 2\text{x}10^{12}$
ReLU	10 x 10	229 ± 32	1138 ± 322
ReLU	25 x 25	422 ± 176	$3\text{x}10^{9} \pm 1\text{x}10^{9}$
ReLU	50 x 50	550 ± 156	$4\text{x}10^{14} \pm 2\text{x}10^{14}$
tanh	10 x 10	253 ± 22	1353 ± 243

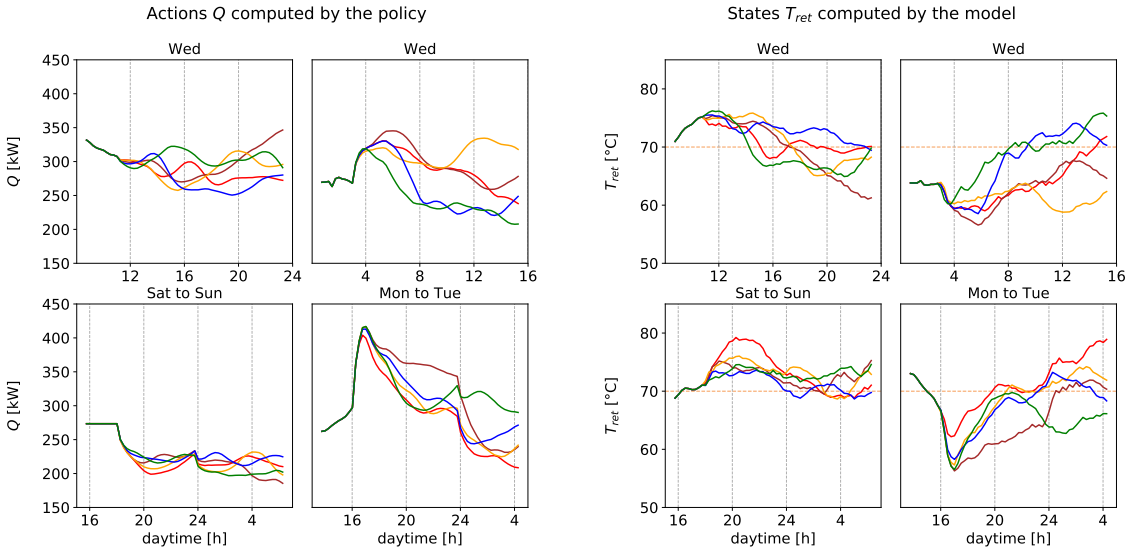
Table 4.1: Performances of several Bayesian neural network configurations with respect to the error metrics introduced in Sec. 4.1.2. Each configuration is trained three times and the results are averaged.

Bayesian Neural Network Configurations

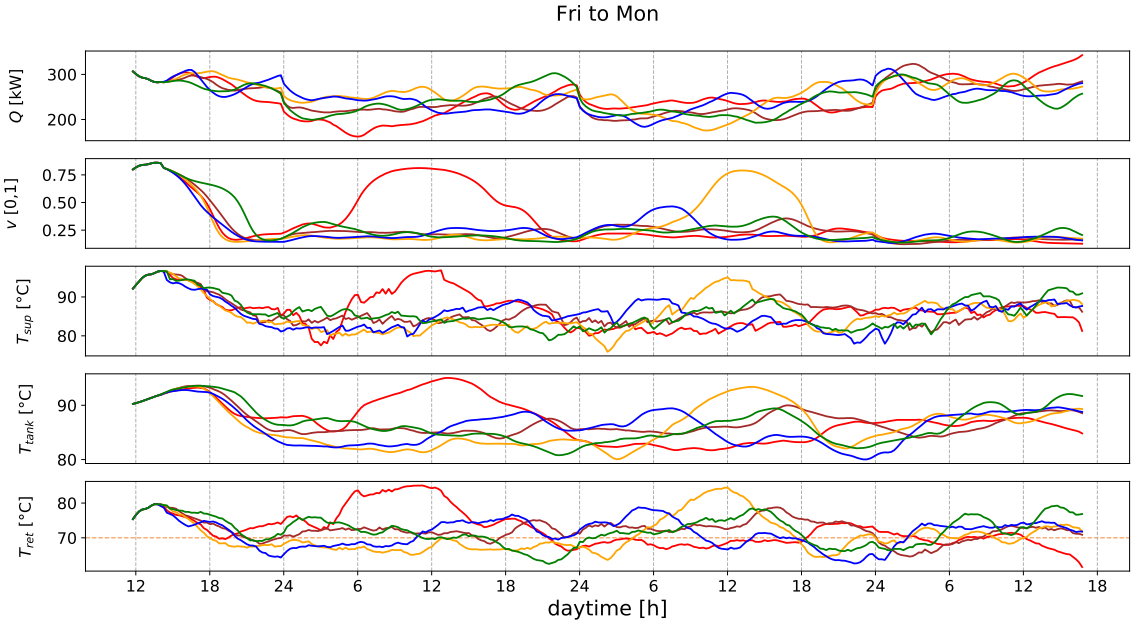
To evaluate the performances of different configurations of Bayesian neural networks, an error metric for model-generated rollouts is introduced. The metric evaluates the accuracy of short-term forecasts with $H = 50$, which corresponds to a forecast horizon of 12.5 hours, and long-term forecasts with $H = 300$, which corresponds to 75 hours. The accuracy measures for short- and long-term forecasts follow the same procedure and are denoted as MSE-S and MSE-L, respectively. For N sequences of actions $\mathbf{a}$ to roll out system and model over a horizon H, the MSE-based metrics are given as

$$E = \frac{1}{N}\sum_{n=0}^{N}\frac{1}{H}\sum_{t=0}^{H}\frac{1}{K}\sum_{k=0}^{K}||\mathbf{s}_{t,k}^{(n)} - \mathbf{s}_{t}^{(n)^*}||^2, \tag{4.1}$$

where $\mathbf{s}$ is a state generated by the model and $\mathbf{s}^*$ is the true state generated by the system. With MSE-S, the accuracy of model-rollouts over a short horizon, as used during policy search, is evaluated. For judging the accuracy and the stability of long-term rollouts, that is, the accumulation of errors over time, MSE-L is used. The performances, according to these metrics, of different configurations of Bayesian neural networks are detailed in Tab. 4.1. The configuration with two hidden layers of 10 ReLU-neurons each scores the lowest error in both categories. Dividing the MSE-S by the dimension of $\mathbf{s}$ and taking the square root gives 8.8°C. Hence, for each state variable s, the average error per time step is around 8.8°C for a horizon of $H = 50$. A Bayesian neural network with 10x10 hidden tanh-units scores slightly higher MSEs. While the configuration of the smallest dimension, one hidden layer with 30 units, scores a relatively low MSE-S, the error on long-horizon forecasts is relatively high. An inspection of the generated rollouts yields that prediction errors accumulate over time, thus the rollouts drift away from the actual system's development. The largest Bayesian neural network with 50x50 hidden units returns the worst score for both metrics. An inspection of the generated long-horizon rollouts yields that their variance is significant and large errors accumulate over time.



(a) Actions Q and states T_{ret} for four samples during policy search. $K = 5$ and $H = 50$.



(b) $K = 5$ rollouts $\mathbf{X}_{P+H}$ with $H = 300$, generated by applying the control policy to the model.

Figure 4.4: Rollouts of the system development generated by the learned model during (a) and after (b) optimization of a control policy π. For each sample, curves with the same color belong to the same rollout.

4.1.3 Policy Optimization Based on the Learned Model

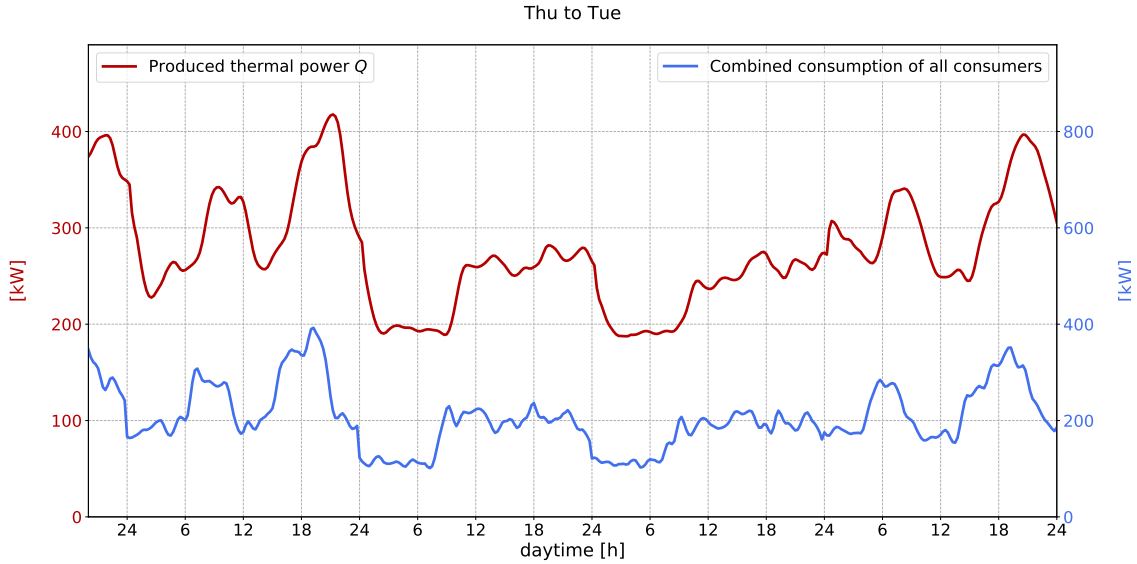
As the Bayesian neural network with 10x10 hidden ReLU-units approximates the dynamics of the district heating system with the highest accuracy, this configuration is used as model for the optimization of a control policy π.
The policy, which is parameterized as a deterministic neural network, is optimized by minimizing (3.5), as detailed in Alg. 1. The desired return temperature $T_{\text{ret,aim}}$ is set to 70°C. The deterministic neural network is pre-trained, and K is initialized as 5 and increased by 1 every 150 training epochs i, as detailed in Sec. 3.2.5.
During training, the number of process sequences $\mathbf{X}_P$ to generate rollouts from, N as in line 7 of Alg. 1, is set to 50. The weight updates in line 19 are calculated by Adam, parameterized as given in Sec. 3.2.5, and the policy is optimized for $I = 1500$ epochs.

As detailed in line 9 of Alg. 1, the policy is optimized by having the learned Bayesian neural network model and the control policy interact to generate rollouts starting from a process sequence $\mathbf{X}_P$. Therefore, process sequences $\mathbf{X}_P$ are referred to as starting sequences in the following.
Rollouts generated by the model during and after optimization of the control policy are provided in Fig. 4.4. The control policy π used to generate the depicted rollouts is a deterministic neural network with 20 hidden units that activate as tanh-functions. In Subfig. 4.4a, sequences of the thermal power Q, computed by the control policy, and the water return temperature T_{ret}, computed by the model, are shown for four different starting sequences $\mathbf{X}_P$. According to the location in the grid-like arrangement, plots of Q on the left side correspond to plots of T_{ret} on the right side. For every sample, $K = 5$ rollouts are shown over a horizon of $H = 50$. For the same sample, identically colored curves are part of the same emerging process sequence $\mathbf{X}_{P+H}$. The control aim $T_{\text{ret,aim}}$ is marked as a dotted line at 70°C. Subfigure 4.4b shows all actions $\mathbf{a}$ and states $\mathbf{s}$ in the process sequence $\mathbf{X}_{P+H}$, generated by having the model and the control policy interact for $H = 300$ time steps. $K = 5$ rollouts are generated and the horizon covers the period between Friday noon and Monday afternoon. For many rollouts in Fig. 4.4, it can be observed that the curves of Q essentially mirror the curves of T_{ret}, or vice versa. This strong correlation is coherent with the real system dynamics depicted in Fig. 1.6. In the individual plots of Subfig. 4.4a, the rollouts deviate from each other, despite originating from the same starting sequence $\mathbf{X}_P$, due to the stochasticity of the Bayesian neural network. In the bottom right sample, T_{ret} shows a rapid decline after 4 pm. Similarly, T_{ret} of the top left sample declines on average after 4 pm. This matches the start of the consumption high in the afternoon and evening, as depicted in Fig. 1.5a. Also for the bottom right sample, a massive rise of the thermal power Q is computed by the control policy at 4 pm in response to the decline of T_{ret}. Eventually, around 5 pm, the temperature T_{ret} is stabilized. This time period is shorter than the given time delay of 70 minutes in the real system.
The cost function, as detailed in Sec. 3.2.4, most notably constrains the thermal

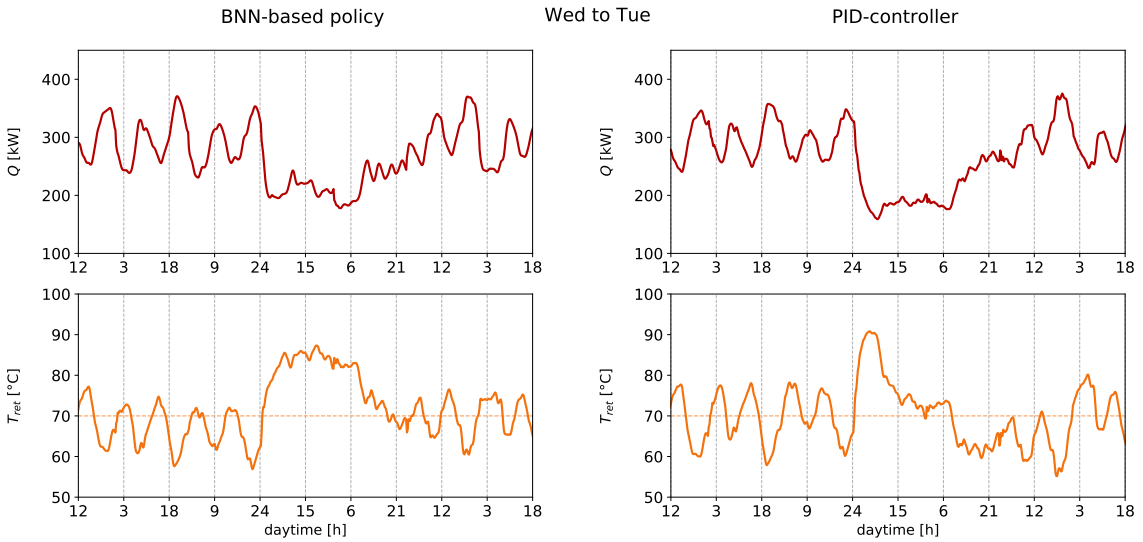
power Q to an interval, and punishes big incremental changes dQ and dv as well as deviations of T_{ret} from $T_{\text{ret,aim}}$. As Subfig. 4.4a shows rollouts generated at a late stage of the policy optimization, indicators for the cost minimization are visible. For the two samples on the left, the control policy keeps the variation of Q relatively low while T_{ret} on average comes closer to $T_{\text{ret,aim}}$ over time. In the rollouts generated for the two samples on the right, the control policy has to find a trade-off between quickly shifting T_{ret} towards $T_{\text{ret,aim}}$ and low incremental changes dQ. However, as shown in Fig. 3.2, a deviation dT_{ret} causes a significant cost. At 5 pm, dT_{ret} is around 0.2 %, which approximately corresponds to a cost value of $c_{dT_{\text{ret}}} \approx 700$. Increasing the thermal power from 300 kW to 350 kW, which corresponds to $dQ = 0.1\bar{6}$ %, causes a cost value of $c_{dQ} \approx 275$. Therefore, large incremental steps of Q are favored by the control policy, and the rise of Q is computed.
In Subfig. 4.4b, rollouts that are generated after optimization are shown. The incremental changes in Q and v are smooth, except for the 24h-marks, and T_{ret} is on average kept close to $T_{\text{ret,aim}}$. In addition, the average thermal power production Q is perceivably lower for Saturday and Sunday than for Friday and Monday. The abrupt changes in Q at day changes are coherent to the real system and stem from the fact that there is a discontinuity induced due to the day-wise sampling of the consumption curves. The fact that the required thermal power Q is less for weekend days is consistent with the consumer models, as the bimodality on average accounts for a lower aggregate consumption on weekends than on weekdays.
In Subfig. 4.4b, T_{sup}, v and T_{tank} show a similar trend. This is coherent with the system rollout depicted in Fig. 1.6. However, further tests show that T_{tank} also correlates with T_{sup} when the valve v is closed, which does hold in the real system.

4.1.4 Optimized Control Policy

Results obtained when applying the optimized control policy π to the district heating system are shown in Fig. 4.5. The depicted control policy is a deterministic neural network with 20 hidden tanh-units. In Subfig. 4.5a, the computed thermal power Q is graphed, in comparison to the aggregate customer demand, over approximately four days. To be able to compare the chronology of peaks and troughs in both curves, the two curves are scaled differently, as indicated by the respective y-axes. It is visible that the two curves follow the same trend. For Friday and Monday, the curves show strong fluctuations in comparison to the time during the weekend.
In Subfig. 4.5b, the control policy π is visually compared to the PID-controller, which is used to generate the dataset for training the model, as detailed in Sec. 3.2.1. For both evaluated control policies, the computed thermal power Q and the response by the system in the form of the state variable T_{ret} are depicted within the same time intervals. The consumption curves and other normally present stochastic effects are generated beforehand, so that both system simulations are identical apart from the applied control policy. The shown interval spans the time from Wednesday until Tuesday.



(a) Produced thermal power Q compared to the aggregate consumption of all consumers.



(b) Comparison of the derived control policy π and the PID-controller. $T_{\text{ret,aim}}$ marked at 70°C.

Figure 4.5: Performance of the control policy π on the district heating network. Subfigure (a) sets the computed thermal power Q in relation to the aggregated customer demand. Subfigure (b) compares the control policy with the PID-controller by showing a sequence of the control input Q and the resulting state T_{ret} for both controllers.

In Subfig. 4.5b, the control inputs specified by the optimized control policy π follow a similar trend as the control inputs by the PID-controller. The same holds for the system response T_{ret}. For the days before the weekend, T_{ret} is on average kept

below $T_{\text{ret,aim}}$, marked as a dotted line, by the control policy π. Oscillations due to morning and evening consumption highs are visible. Subsequently, the aggregate consumption on the weekend drops because of the bimodal consumer behavior and as a consequence, T_{ret} rises and Q decreases. With the start of Monday, the morning and evening oscillations ensue again.
In Subfig. 4.5b, a predictive behavior is observable for both the control policy π and the PID-controller, just before the 24h-mark between Friday and Saturday. Even though T_{ret} has not yet reached the control aim $T_{\text{ret,aim}}$, the specified thermal power production Q is already being reduced, as it is apparent that T_{ret} starts to rise.
A difference between the produced sequences of T_{ret} by the control policy and the PID-controller is perceivable for the time interval that spans the weekend days. The difference in the sequences of the specified control input Q is smaller in comparison. This is because during times of low consumption, the response by the system to the control inputs is significantly stronger, which leads to the strong rise of T_{ret}. During the second half of Sunday, the specified Q-curves of both controllers chatter. When the controllers sense a pivot point of T_{ret}, as pointed out earlier, the production of the thermal power Q also takes a pivot, because it is expected that the previous control inputs are sufficient to move T_{ret} to the control aim $T_{\text{ret,aim}}$. However, during a period of small consumptions the time delays of the consumers do not play a significant role, thus the dynamics are different and T_{ret} reacts more quickly to changes of Q. Yet, at the time of the pivot, the specified amount of heat Q is not sufficient to raise T_{ret} to $T_{\text{ret,aim}}$, so T_{ret} drops again, and Q is again increased in turn, which leads to the observable chattering.
To compare the performance of differently configured deterministic neural networks as control policies π, the cumulative cost function (3.5) is used. The cumulative costs of applying the control policies to the model as well as to the underlying system are compared. To compute the cost for the performance on the model, $K = 25$ rollouts over a horizon of $H = 100$ are generated for 500 different starting sequences $\mathbf{X}_P$. For assigning a cost to the performance of steering the system, the policy is applied to the system for 100 days and the cumulative cost (3.5) of the resulting sequence $\mathbf{X}$ is calculated with $K = 1$. The consumption curves and other normally present stochastic effects of the system are generated beforehand and re-used in each simulation, so that all control policies act upon the exact same system. Each configuration is trained twice and the costs are averaged. The results are given in Tab. 4.2.
The control policy that is parameterized as deterministic neural network with 20 hidden tanh-units returns the lowest costs for both the learned model by the Bayesian neural network and the real district heating system. The cost of steering the real system is slightly lower than the cost assigned to the PID-controller. A control policy with 15x15 hidden ReLU-units is assigned a cost similar to the PID-controller's cost. However, in this implementation these configurations suffers from dying ReLUs during optimization, thus are constrained in the computation of actions $\mathbf{a}$ [CUH16].

Type	Activation	Hidden units	Model cost	System cost
Policy	tanh	20	285 ± 7	366 ± 4
Policy	ReLU	15 x 15	345 ± 25	373 ± 17
PID	-	-	-	372 ± 0

Table 4.2: Costs of differently dimensioned control policies π and the PID-controller.

4.2 Discussion

The results show that a Bayesian neural network with stochastic inputs is capable of modelling the district heating system sufficiently accurate so that a control policy, which performs similar to a PID-controller, can be derived. However, this outcome is not satisfactory from a practical point of view, as the design of a PID-controller is by far less complex than the procedure of Bayesian neural network based policy search.
In this section, the benefits and shortcomings of Bayesian neural network based policy search both in general and of this specific implementation are discussed based on the results provided in Sec. 4.1. After discussing the policy optimization process, the performance of the control policy is evaluated. Subsequently, the accuracy of the system representation learned by the Bayesian neural network is discussed. The section closes by detailing the complexity of the used optimization methods and relating insights found in this thesis to the results reported for Bayesian neural network based policy search by a related study [DHLDVU17].

4.2.1 Model Based Policy Optimization

The results show that updates for the parameters $\mathcal{W}_\pi$ of a control policy π based on gradients of the cost that arises by applying the policy to a learned system dynamics model allow for generating a useful controller. Minimizing the cost function by exploring the learned model also leads to a decreased cost when steering the real system. For this setting, it is shown that the derived control policy π performs slightly better with respect to the cost than every other control policy that is applied to the system to generate the dataset for the model training. A cost for comparison is only provided for the PID-controller, as it scores the lowest cost of all four initial policies that are detailed in Sec. 3.2.1. Hence, an existing control policy can be improved with respect to a defined cost function by exploration in a learned dynamics model.
However, as listed in Tab. 4.2, the tanh- policy achieves a significantly lower cost on the model-generated rollouts than on the real system. Even though the model cost of the ReLU-policy is significantly higher than the cost of the tanh-policy, the system costs are almost the same. Hence, the control policy learns to cope with cost-causing effects that are present only in the learned dynamics model. This underlines

the necessity of an accurate dynamics model. If the learned dynamics model is not accurate, then even a control policy that scores a cost of zero on the model rollouts may fail to steer the underlying system well with respect to the cost metric. In conclusion, the achievable cost on the real system is bounded by the accuracy of the learned dynamics model.

4.2.2 Performance of the Control Policy

Even though the obtained control policy is capable of steering the district heating system under a cost that is comparable with the benchmark controller, the desired control properties have not been sufficiently incorporated into the control policy. The control policy fails to compensate peaks of aggregated consumer demand, which might be done by specifying a high amount of thermal power Q in advance, and as a consequence the return temperature T_{ret} fluctuates around the desired value. Nevertheless, a predictive control behavior that is shown by the PID-controller, due to the differential D-part [ACL05], is also incorporated into the optimized control policy π. The D-part of the PID-controller calculates a trend of the control error $T_{\text{ret}} - T_{\text{ret,aim}}$, as given in (3.7). Judging by the similar behavior of the control policy π shown in Sec. 4.1.4, a trend of the deviation of the return temperature T_{ret} is also sensed. The control policy is a deterministic neural network with bias nodes in each layer, as detailed in Sec. 2.1.1. Thus, the trend can be calculated by fitting the bias node in the input layer to act as $T_{\text{ret,aim}}$ and have the hidden neurons subtract it from the previously observed return temperatures T_{ret} in the process sequence $\mathbf{X}_P$, which is given as input.
Even though the cost function is minimized during the policy optimization, the performance of the resulting control policy π is not satisfactory. This might be due to a performance limit imposed by the inaccuracy of the learned system model, as discussed in Sec. 4.2.1. On the other hand, it cannot be ruled out that only a local minimum of the cost function is found during policy search. After 1,500 epochs, the cost does not drop significantly anymore. However, the parameters of the policy are updated by Adam, which incorporates momentum [KB14]. Thus, the control policy might escape the local minimum, if the training is further continued. Increasing the number of training samples N and the length of the rollout horizon H may minimize the cost further. Yet, it is unclear whether a further decreased model cost necessarily implies a lower system cost, as pointed out in Sec. 4.2.1.

4.2.3 Performance of the Bayesian Neural Network

In the following, the performance of the Bayesian neural network as dynamics model is discussed. After detailing the general performance of the system representation, specific properties are detailed by analyzing the results. Subsequently, remarks about the model's representation of the stochastic effects, which are present in the

district heating network, are made. Finally, the dimensioning of the hidden layers of the Bayesian neural network is discussed.

Learned System Representation

The benchmark district heating network is a system that is difficult to model. The time delays are significant, as it takes 70 minutes before the specified thermal power Q has an impact on the return temperature T_{ret}. In addition, the system exhibits a strong stochasticity due to the attached consumers, whose consumptions vary for each weekday and are bimodal during weekends. Yet, as a stable control policy is derived, the system representation learned by the Bayesian neural network is true to some extent. However, the large errors listed in Tab. 4.1 and the discrepancies between model-generated and real rollouts, graphed in Fig. 4.2, reveal the inaccuracies of the learned system representation.
The results in Sec. 4.1.2 show that characteristics of the real system are replicated by the learned model at times. The influence of the trend of T_{ret} on the trend of T_{sup} is represented by the model, however not always accurately as shown in the right column of Fig. 4.2a for the sample of the P-controller. The variables T_{sup}, v and T_{tank} in model-generated rollouts show similar correlations as their real counterparts in Fig. 1.6. Yet, T_{sup} and T_{tank} inaccurately show correlations also if the valve v is closed. The model replicates the decrease of T_{ret} due to the afternoon consumption peaks at times, but the impact of the consumption peaks in the early morning is hardly represented. Nevertheless, a tendency of a lower aggregate consumption on weekends is learned by the model. The essential relationship between rising thermal power Q and rising return temperature T_{ret} is incorporated into the model and the learned policy. Nevertheless, factors that constrain the control input Q to an interval have to be incorporated into the cost function (3.9), as the model falsely tends to generate constant temperatures T_{ret} for relatively low constant control inputs Q. These observations lead to the conclusion, that the learned model replicates the characteristics of the system only in local state and action spaces.
It is experimentally determined that $P = 9$ suffices for the process sequence $\mathbf{X}_P$ to contain all relevant past information to cope with the time delays present in the system. For this reason, and drawing on remarks made in Sec. 1.2.2 about the learnability of the consumption curves, the 48-dimensional input vector (see Subtab. 3.1b) contains all information that is necessary to learn the system dynamics up to sampling noise in generating the consumption curves. This is, assumed that the amount of training data is sufficient to represent the day-dependent consumption characteristics and that the control policies accurately depict the relation between actions $\mathbf{a}$ and states $\mathbf{s}$. Nevertheless, the high dimensions of the input and the output makes it a difficult task for the optimization algorithm to determine the correct relationships and fit the weights $\mathcal{W}$ correctly, as elucidated in the following. The rollouts in Fig. 4.4a, most of all the bottom right sample, show that the Bayesian

neural network learns an anti-causal relationship between Q and T_{ret}. This means that whenever a strong incremental change dQ appears, the model assumes that the return temperature is moving in the opposite direction and predicts a corresponding change of T_{ret}. This effect is induced by the P- and PID-controllers, which are used as control policies for the dataset generation. Whenever aggregate consumptions peak, the water temperatures in the pipes drop. This in turn causes either controller to produce more thermal power Q to compensate for the peak demands, and based on that the Bayesian neural network falsely infers that a rising Q implies a falling T_{ret}, and vice versa. This happens twice per day, for the morning and evening times of peak consumption, as seen in the Figures 1.5a and 3.1. The sinusoid control policy, introduced in Sec. 3.2.1, mitigates but does not resolve this effect.

Modelling Stochasticity with Bayesian Neural Networks

It is shown in Fig. 4.1 that Bayesian neural networks with stochastic inputs are capable of modelling the bimodality and the heteroscedasticity exhibited by the consumers. The results provided in Sec. 4.1.2, specifically in Fig. 4.3 show that these effects are also replicated in the learned model of the district heating system by the Bayesian neural network with 50x50 hidden units. This is remarkable for two reasons. On the one hand, the district heating system serves six customers who, on weekend days, exhibit a bimodal consumption behavior, independently of each other. This accounts for seven modes of the aggregate consumption of all consumers. As each consumer c features different shapes of the bimodal curves due to the consumer- and day-dependent characteristics $\mathcal{C}_d^c$ in (1.7), and sampling noise further perturbs the consumption curves, as detailed in Sec. 1.2.2, the seven emerging modes are expected to be not clearly separable anymore. On the other hand, in the isolated setting depicted in Fig. 4.1, the Bayesian neural network receives one input and can optimize all its parameters to generate one output. When modelling the benchmark system, the Bayesian neural network has to optimize a mapping from an input vector that consists of 48 variables to the output vector $\mathbf{s}_{t+1}$ that consists of 3 variables. In the input vector, the only direct indicator for multimodality is the variable $d \in [0, ..., 6]$, which gives the day of the week. As in the isolated setting, the model also receives the daytime τ to infer the magnitude of the consumption. Drawing on these two variables, the weights have to be fitted so that day-dependent heteroscedasticity and bifurcation points on weekend days are modelled, in addition to learning the influence of the control variables $\mathbf{a}$ on the state $\mathbf{s}$.

Dimensioning the Bayesian neural network

Even though the Bayesian neural network with 50x50 hidden units reproduces the multimodality induced by the weekend consumptions to some extent, the variance in the predictions is large and prediction errors are accumulatated over time. This is

listed in Tab. 4.1 and perceivable by comparing the ticks of the y-axes of the 10x10 model's rollouts in Subfig. 4.2a and the 50x50 model's rollouts in Fig. 4.3. Therefore, even though a larger number of hidden units benefits the modelling of complex stochastic effects, the variation in the predictions is generally stronger. Thus, large errors are accumulated in a setting like model-based policy search, where the Bayesian neural network is iteratively applied to its own predictions. On the other hand, even though a smaller Bayesian neural network with 10x10 hidden units has less dimensions to model complex stochastic effects, the predictions generally have a lower MSE and rollouts are more robust with respect to error accumulation.

4.2.4 Results in the Literature and Training Complexity

Bayesian Neural Network Performance

Depeweg et al., the authors of the publication in which Bayesian neural network based policy search is proposed, reported results of a Bayesian neural network with 75x75 hidden units in one of their experiments [DHLDVU17]. Results for Bayesian neural networks with less hidden units are not reported for the respective task, but it is assumed that the authors evaluated different dimensions and the 75x75 configuration performs best. Therefore, it is surprising that the Bayesian neural network with 10x10 hidden units performs best in this thesis. For the aforementioned experiment conducted by Depeweg et al., the input vector is a sequence of past states and actions, similar to the process sequence $\mathbf{X}_P$ used in this setting. The input comprises 60 variables compared to 48 in this study. The output is one-dimensional while it is the three-dimensional state $\mathbf{s}$ in this setting. 70,000 training samples are used by Depeweg et al., while 135,000 are used for learning the dynamics of the district heating network. Hence, the reason for both the 25x25 and the 50x50 model performing worse than the 10x10 model is not totally clear, but possible hypotheses are discussed in the following. The stochasticity caused by the consumers is complex but learnable up to some extent, as elucidated in Sec. 1.2.2. Therefore, the 50x50 model might focus on fitting the complex effects caused due to varying consumptions while missing the essential dynamics of the district heating network. A partial fit of the stochastic consumption behavior at the expense of a large MSE is pointed out in the previous paragraph and backs this hypothesis. On the other hand, the simpler 10x10 configuration might do the opposite what leads to a better MSE. Another possible reason is that simulating the district heating network does not require as many hidden units as the problem investigated by Depeweg et al., which is learning a mapping in a dataset generated by an industrial benchmark simulator. However, this seems improbable. If it were the case, the rollouts generated by the Bayesian neural network with 10x10 hidden units, which are depicted in Fig. 4.2, should be more accurate.

Training Bayesian neural networks with BB-α Training a relatively small Bayesian neural network with 10x10 hidden units for 20,000 epochs on the dataset detailed in 4.1.2 takes roughly 60 hours with an Intel dual core i7-6500 CPU @ 2.50 GHz. However, this process can be sped up by leveraging the parallelization capabilities of a GPU. In addition, the parameter updates are calculated using batch-gradients. A formulation of the energy function (2.21) tailored to optimization by minibatch gradient-descent [Rud16] speeds up the minimization and is provided in [DHLDVU17]. Using minibatch-gradients and a GPU, a duration of 5 hours is reported for training a Bayesian neural network with 75x75 hidden units on a dataset that consists of 70,000 pairs of 60-dimensional samples and 1-dimensional labels.
Moreover, minimizing the energy function (2.23) is not always straightforward. It may happen that the log-likelihood term becomes zero if the prediction strongly differs from the target, leading to a log(0), which is undefined. Furthermore, the factors $f(\mathcal{W})$ and $f(z_n)$ (2.19) contain denominators with a difference between the prior variance and the variance v that is to be fitted. If v approaches the prior during optimization, the denominator becomes small. In turn, the fractions becomes large. The exponentiation of the large fraction may lead to a value that exceeds the maximum value that can be represented with a float of 64-bit precision.
Finally, the theory of BB-α is not trivial. This makes it difficult to fully understand what happens during the minimization of the energy function. Hence, trying to find reasons for poor results in the optimization of the parameters is a tedious process and requires a profound knowledge of statistics.

Training Complexity of Bayesian Neural Network Based Policy Search

It is reported that, using a Bayesian neural network with 75x75 hidden units that takes a 60-dimensional input vector as model, optimizing a deterministic neural network with 75x75 hidden ReLU-units takes 14-16 hours on a CPU, for 50 samples over a horizon $H = 75$ with $K = 25$ rollouts per sample [DHLDVU17]. In this implementation, optimizing a 10x10 deterministic neural network with 20 hidden tanh-units for 50 samples and a horizon of $H = 50$ over 1500 epochs takes roughly three days with an Intel dual core i7-6500 CPU @ 2.50 GHz, even though the adaptive-K method that is detailed in Sec. 3.2.5, is applied to speed up the policy search process. The policy search process may be sped up by a GPU-compatible implementation.

Chapter 5

Conclusion

Despite its complexity and stochasticity, the district heating system is modelled by the Bayesian neural network with a sufficient accuracy, so that a stable control policy is derived. The control policy performs similarly to the PID-controller that is applied to the district heating system to generate the dataset for training the Bayesian neural network. The obtained control policy slightly improves upon the cost of the PID-controller. Hence, it is shown that an existing control policy can be optimized by Bayesian neural network based policy search. Nevertheless, it is not achieved to incorporate desirable properties into the control policy, like foreseeing recurrent times of peak consumption and compensating for these in time, so that the temperature of the water returned by the distribution network is kept stable.

It is shown that the Bayesian neural network with stochastic inputs partially captures multimodalities and heteroscedasticity present in the system, despite these being indicated by just one variable in the 48-dimensional input vector. However, replicating these effects requires a Bayesian neural network with a relatively large number of hidden units, which in this setting produces a significantly larger modelling error than a network with less hidden neurons.
A closer validation of the system approximation learned by the Bayesian neural network reveals inaccuracies. Efforts are made to generate an expressive dataset for training, and the Bayesian neural network partially reproduces characteristics of the system when it is given data similar to that observed during training. On the other hand, fundamental characteristics of the system, like the influence of the produced thermal power on the return temperature, are not accurately replicated in action and state spaces that are not covered by the training data. This highlights a barrier in learning a dynamics model by statistical optimization. With statistics, the model's parameters are fitted so that the coherences present in a dataset are represented well according to an optimality criterion, like a loss function. Yet, the learned model lacks a deeper understanding of the processes that are at work in the system and when confronted with an unexplored portion of the system space, calculations that apply to known system spaces are performed to obtain a prediction.

In this thesis, the influence of hyperparameters during policy search, like the number of samples and the length of the rollout-horizon, is not sufficiently explored. However, the Bayesian neural network has difficulties to learn a dynamics model that generalizes to unseen state spaces of the system. This raises the question whether a control policy with desirable properties can be obtained based on a Bayesian neural network model, if these properties are not to some extent present in the control policies that are used to generate the dataset for training this model.

Outlook As pointed out, a statistically optimized dynamics model may perform poor in state spaces that are not covered by the training dataset. As an advantage over deterministic models, a Bayesian neural network, whose parameters are described by probability distributions, additionally allows for judging the confidence that is attached with its output. Therefore, the rollouts generated during Bayesian neural network based policy search may be assigned a cost according to a variation-based metric. This way, the control policy is automatically prevented from exploring spaces in which the predicted rollouts are uncertain.
As explained in Sec. 4.2.1, it is not clear whether a further minimization of the cost function by exploring the learned dynamics model results in a better control policy. Nevertheless, this is not evaluated and the cost may be further minimized by training the policy with gradients calculated from rollouts over a longer horizon. In addition, the amount of samples may be increased to obtain more expressive gradients that hold for a larger state- and action-space of the learned dynamics model, and extract valuable information for parameter updates despite the stochasticity that comes with the rollouts of the Bayesian model.
The performance of the Bayesian neural network might be enhanced by pruning the 48-dimensional input vector. As the time delays are known in this setting, the Bayesian neural network may be prevented from learning the anti-causal relationship described in Sec. 4.2.3 by excluding the most recent actions $\mathbf{a}$ from the input. The impact of the stochastic consumer behavior may be mitigated by an even larger dataset or using seven Bayesian neural networks due to the day-dependent variation of the consumptions. This way, the system's varying stochasticity does not need to be inferred by one of several input variables but is steady for each model.
Furthermore, a more accurate system representation might be obtained by extending the Bayesian neural network model with a prediction mechanism for the external temperature trend, which is assumed constant in this implementation.
The evaluated procedure optimizes a control policy by minimizing a cost that is averaged over a specified number of expected system rollouts that are generated by the Bayesian neural network. However, the capability of the Bayesian neural network to model complex stochastic effects might also be leveraged for deriving risk-averse control policies in safety-relevant settings, by focusing on worst-case rollouts that lead to hazardous scenarios.

List of Figures

Acronyms

BB-α	black-box α-divergence minimization.
MAP	maximum a posteriori estimation.
MLE	maximum likelihood estimation.
MPC	model-predictive control.
MSE	mean squared error.
P	proportional.
PID	proportional integral derivative.
ReLU	rectified linear unit.
tanh	tangens hyperbolicus.

Notation

x	scalar with the name x.
$\mathbf{x}$	vector with the name x.
$\mathbf{X}$	matrix with the name X.

Bibliography

[ACL05] K. H. Ang, G. Chong, and J. Li. PID control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology*, 2005.

[BCD+19] S. Buffa, M. Cozzini, M. D'Antoni, M. Baratieri, and R. Fedrizzi. 5th generation district heating and cooling systems: A review of existing cases in Europe. *Renewable and Sustainable Energy Reviews*, 2019.

[Ber08] D. P. Bertsekas. *Approximate Dynamic Programming.* MIT Press, 2008.

[BKPW13] P.S. Booji, V. Kamphuis, O.P. van Pruissen, and C. Warmer. Multi-agent control for integrated heat and electricity management in residential districts. *4th International Workshop on Agent Technologies for Energy Systems*, 2013.

[BT95] P. T. Boggs and J. W. Tolle. Sequential quadratic programming. *Acta Numerica*, 1995.

[CUH16] D. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *International Conference on Learning Representations*, 2016.

[CVDR18] B. Claessens, D. Vanhoudt, J. Desmedt, and F. Ruelens. Model-free control of thermostatically controlled loads connected to a district heating network. *Energy and Buildings*, 2018.

[DHDVU18] S. Depeweg, J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft. Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning. In *International Conference on Machine Learning*, 2018.

[DHLDVU17] S. Depeweg, J. Hernández-Lobato, F. Doshi-Velez, and S. Udluft. Learning and policy search in stochastic dynamical systems with bayesian neural networks. *International Conference on Learning Representations*, 2017.

[DR11] M. P. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning*, 2011.

[DS14] N. Draper and H. Smith. *Applied Regression Analysis, Third Edition*. Wiley Series in Probability and Statistics, 2014.

[FCB07] E. F. Camacho and C. Bordons. *Model Predictive Control*. Springer, Berlin, Heidelberg, 2007.

[Gal16] Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[GTS+16] V. Gómez, S. Thijssen, A. Symington, S. Hailes, and H. Kappen. Real-time stochastic optimal control for multi-agent quadrotor systems. *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling*, 2016.

[HLR+16] J. M. Hernández-Lobato, Y. Li, M. Rowland, T. Bui, D. Hernández-Lobato, and R. Turner. Black-box alpha divergence minimization. In *International Conference on Machine Learning*, 2016.

[ISK+14] E. Ikonen, I. Selek, J. Kovács, M. Neuvonen, Z. Szabó, J. Bene, and J. Peurasaari. Short term optimization of district heating network supply temperatures. *IEEE International Energy Conference*, 2014.

[Joh14] C. Johansson. *On intelligent District Heating*. PhD thesis, Blekinge Institute of Technology, 2014.

[Kap07] H. Kappen. An introduction to stochastic control theory, path integrals and reinforcement learning. *9th Granada seminar on Computational Physics: Computational and Mathematical Modeling of Cooperative Behavior in Neural Systems*, 2007.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Computing Research Repository*, 2014.

[KKN11] E. Kreyszig, H. Kreyszig, and E. J. Norminton. *Advanced Engineering Mathematics*. Wiley, tenth edition, 2011.

[KSH12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012.

[KSW15] D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. In *International Conference on Neural Information Processing Systems*. MIT Press, 2015.

[Lev17] F. Levihn. CHP and heat pumps to balance renewable power production: Lessons from the district heating network in Stockholm. *Energy*, 2017.

[LHP+15] T. P. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *Computing Research Repository*, 2015.

[MDA15] D. Maclaurin, D. Duvenaud, and R. P. Adams. Autograd: Effortless gradients in numpy. *International Conference on Machine Learning*, 2015.

[Min05] T. Minka. Divergence measures and message passing. *Microsoft research*, 2005.

[Moo65] G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, 1965.

[Nea96] Radford M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1996.

[Ng04] A. Ng. Feature selection, L1 vs. L2 regularization, and rotational invariance. In *International Conference on Machine Learning*, 2004.

[NH10] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*. Omnipress, 2010.

[Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65, 1958.

[Rud16] S. Ruder. An overview of gradient descent optimization algorithms. *Computing Research Repository*, 2016.

[SAB14] H. Sak, Senior A., and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. *Interspeech*, 2014.

[Sch93] J. Schmidhuber. *Netzwerkarchitekturen, Zielfunktionen und Kettenregel*. PhD thesis, Technical University Munich, 1993.

[Sch15] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 2015.

[SFT+04] G. Sandou, S. Font, S. Tebbani, C. Mondon, and A. Hiret. Global modelling and simulation of a district heating network. *International Symposium on District Heating and Cooling*, 2004.

[SFT+05] Guillaume Sandou, Stéphane Font, Sihem Tebbani, Arnaud Hiret, and Christian Mondon. Predictive control of a complex district heating network. *IEEE Conference on Decision and Control*, 2005.

[SHK+14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 2014.

[SHM+16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016.

[SLZL16] L. Sun, D. Li, Q. Zhong, and K. Y. Lee. Control of a class of industrial processes with time delay based on a modified uncertainty and disturbance estimator. *IEEE Transactions on Industrial Electronics*, 2016.

[TBS10] E. A. Theodorou, J. Buchli, and S. Schaal. A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research*, 2010.

[WAT15] G. Williams, A. Aldrich, and E. A. Theodorou. Model predictive path integral control using covariance variable importance sampling. *Computing Research Repository*, 2015.

[Wer74] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974.

[Wer17] S. Werner. International review of district heating and cooling. *Energy*, 2017.

[XWCL15] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *Computing Research Repository*, 2015.

[Zel94] A. Zell. *Simulation Neuronaler Netze*. Oldenbourg, 1994.

License