Ludwig-Maximilians-University Munich

Meteorological Institute

Master Thesis

Combining Data Assimilation and Machine Learning to Estimate Parameters of a Convective-Scale Model

Author:
Stefanie Legler

Supervisor:
Tijana Janjic Pfander

June 18, 2021

Ludwig-Maximilians-Universität
München

Meteorologisches Institut

Masterarbeit

Kombination von Datenassimilation und Maschinellem Lernen zur Parameterschätzung eines Konvektiven Modells

Autorin:
Stefanie Legler

Betreuerin:
Tijana Janjic Pfander

June 18, 2021

Combining Data Assimilation and Machine Learning to Estimate Parameters of a Convective-Scale Model

Stefanie Legler

Abstract

Numerical weather prediction models are subject to parametrizations of subgrid and complex physical processes such as cloud development in convective scale models. Traditionally, the numerical values of these model parameters are chosen either by manual model tuning or, more objectively, by the augmented state approach during the data assimilation. In this thesis, we look at the problem of parameter estimation through an artificial intelligence (AI) lens by training two types of artificial neural networks to estimate several parameters of the one-dimensional modified shallow-water model as a function of the atmospheric state obtained by observations or the stochastic Ensemble Kalman Filter. Through perfect model experiments, we show that Bayesian neural networks as well as Bayesian approximations of points estimate neural networks can estimate model parameters with relevant statistics and manage to decrease the initial state errors even under sparse and noisy conditions. The sensitivity to the number of ensemble members, observation coverage, and neural network size is shown. Additionally, we use the method of layer-wise relevance propagation to gain insight into how the artificial neural networks are learning and discover that they naturally select only a few grid points that are subject to strong winds and rain to make their predictions.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Correctly predicting the weather can help politicians to evacuate flood areas in time, utility companies to provide enough resources when needed, and the average Joe to not get caught in the rain without an umbrella. The earliest attempts of forecasting the weather reach back thousands of years when people used optical phenomena such as the color and shape of clouds or the occurrence of halos around the moon or sun to predict short-term changes in the atmosphere (Frisinger, 1978). The birth of modern meteorology is attributed to Robert FitzRoy in 1860, who used the newly established telegraph network to gather daily, local weather reports at set times (Mellersh, 1968). Together with barometers, records of atmospheric patterns, and his own nautical charts he was able to make the first primitive synoptic analyses and published daily weather forecasts in *The Times* in 1861. It took humanity another half of a century to lay the foundations of modern numerical weather prediction.

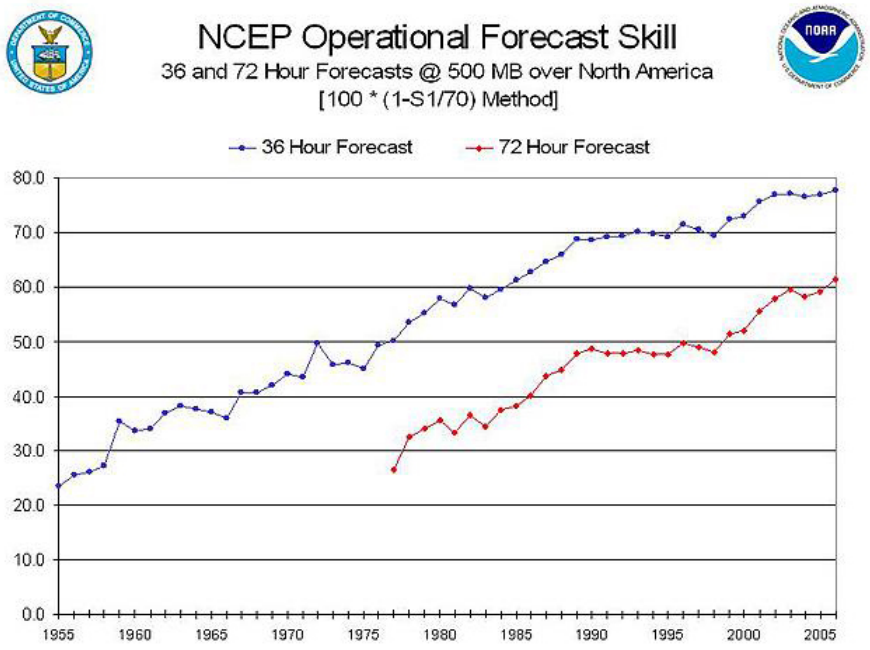

Figure 1.1: Operational Model forecast skill scores from 1955 - 2006 provided by the National Centers for Environmental Prediction (NCEP)

The first practical effort of simulating the future change of the current state of

the atmosphere using mathematical models was attempted in the 1920s by Lewis Fry Richardson (1922) based on theoretical work by Abbe (1901) and Bjerknes (1904). Richardson's six-hour forecast of the change in surface pressure over two points in Central Europe was computed manually for 6 weeks and predicted values that were incorrect by two orders of magnitude. With advances in computational power and atmospheric physics, weather prediction became operational in 1955 and steadily increased its forecast skill since then (Figure 1.1). The large error in Richardson's first attempt was attributed to an imbalance in the wind velocity and pressure fields, that were used as the initial conditions. This shows that choosing the right initial conditions is not a trivial task and emphasizes the importance of data assimilation (DA) as its own research field. DA developed from simple interpolation approaches to the now widely used methods of combining observations with *a priori* computed atmospheric states using state-of-the-art numerical weather prediction models. By including error covariances of observations and numerical forecasts DA is able to not only compute the initial state of the atmosphere but its uncertainty as well. However, a large contribution to the numerical weather prediction model error is due to the parametrization of microphysics and due to processes in the surface and boundary layers. Estimating these tunable parameters and allowing for uncertainty in them can lead to a more realistic respresentation of the model error (Ruckstuhl and Janjić, 2018). A detailed overview of modern DA methods and parameter estimation in DA will be given in Section 2.1.

In recent years machine learning (ML) has become a subject of interest in various research fields within atmospheric physics. Attempts of including ML into the cycle of climate and weather modeling reach from using ML to represent sub-grid processes in global climate models (O'Gorman and Dwyer, 2018; Rasp et al., 2018; Yuval and O'Gorman, 2020), over replacing DA by an artificial neural network (ANN) to emulate an Ensemble Kalman Filter (EnKF) (Cintra and Campos Velho, 2014), to utilizing an ANN as a surrogate for the dynamical model of the atmospheric state during the DA (Brajard et al., 2020; Ruckstuhl et al., 2021). Although the ML algorithms show promising results in these idealized test cases, they come with two major drawbacks. On one hand, ANNs typically do not provide an uncertainty with their predictions, which makes it hard to ascribe a confidence when using ANNs in operational settings. On the other hand, they are still seen as black boxes that do not provide any insight into the functions they are trying to approximate. To tackle the latter problem Toms et al. (2020) introduced layer-wise relevance propagation (LRP) to the geosciences, which can be used to visualize how the ANN makes its prediction. Labe and Barnes (2021) utilized this method to disentangle relative influences on regional surface temperatures of aerosols and greenhouse gases in the atmosphere. The former problem could be approached by using stochastic ANNs instead of their widely used deterministic counterpart.

The goal of this thesis is threefold. First, to estimate parameters of the convective-scale modified shallow water model from sparse and noisy observations using ML and DA. Second, to compare the predictions and statistics of stochastically trained Bayesian neural networks (BNNs) with deterministically trained point estimate neural networks (NNs). And third, to investigate the influence of the model parameters on the model states by applying LRP onto the trained ANNs.

Chapter 2

Theoretical Background

The following chapter summarizes the theoretical basics of the methods developed for this thesis. It is divided into two parts. The first one is concerned with the relevant DA methods, especially the stochastic EnKF. The second part outlines the theory behind ANNs while focusing on the differences between deterministically and stochastically trained ANNs.

2.1 Data Assimilation

Given an initial state of the atmosphere, numerical weather models simulate the evolution of geophysical variables, e.g. potential temperature, wind, pressure. Sources of information about this initial state are observations and usually the most recent forecast from a numerical model (*background*). The procedure of combining these different sources to obtain an estimate (*analysis*) of the true state of the atmosphere is called data assimilation. Using only observations as the initial state is usually not feasible as they are mostly sparse and not made on the same grid points as the numerical model. Moreover, they sometimes measure the atmospheric variables only indirectly and are subject to observational errors, that have to be accounted for. If one would only use a numerical model to make weather predictions without any input from observations, the model would drift away from reality over time due to uncertain initial conditions and due to model error, caused by discretization and approximations of small-scale and complex physical processes, known as parameterizations.

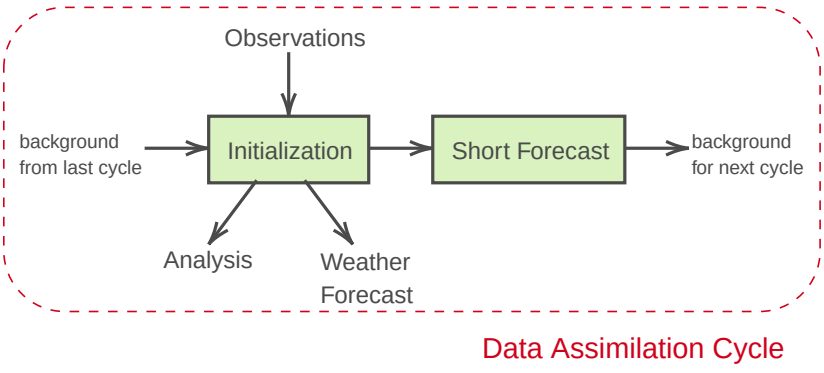

Figure 2.1: Scheme of Data Assimilation Cycle

A *DA cycle* (Figure 2.1) usually consist of two steps. The *analysis step*, where a model state is estimated using observations and the background state. And the *forecast step*, where the analysis is propagated forward in time using the numerical model until the next DA cycle is performed. Most DA methods in use nowadays are based on minimizing a quadratic cost function of model states and differ in their representation of the error statistics, computational cost, and accuracy. Although the parameter estimation algorithms presented in this study can generally be used with any DA method, here they were tested in combination with the stochastic EnKF based on the algorithm described in Evensen (2003) for estimating the state of the atmosphere.

2.1.1 Stochastic Ensemble Kalman Filter

Even though Kalman Filter-based methods assume that the errors of the atmospheric variables are Gaussian distributed, which is often violated on the convective scale, they are still a popular tool for DA due to their affordable computational requirements. One way of dealing with the suboptimal Gaussian assumption is to enforce constraints based on physical conservation laws during the DA (Janjic et al., 2014, QPEns), which results in more physically plausible states for the ensemble mean and the individual ensemble members. In Ruckstuhl and Janjić (2018) this approach was successfully applied on a convective scale model. Since the focus of this work is testing new algorithms for parameter estimation, a simple stochastic EnKF without constraints will be used. It is based on the following cost function for each of the N_{ens} ensemble members:

$$J(x_t^{a,i}) = (x_t^{f,i} - x_t^{a,i})^T P_t^{-1} (x_t^{f,i} - x_t^{a,i}) + (y_t^i - H_t x_t^{a,i})^T R_t^{-1} (y_t^i - H_t x_t^{a,i}) \tag{2.1}$$

where $i, i = 1...N_{ens}$ denotes one ensemble member, $x_t^{f/a,i}$ are the background and analysis states respectively, R_t is the observation-error covariance matrix and H_t denotes the observation operator, that maps the model states to the observation space. In this work we assume H_t to be linear. $\{y_t^i\}$ represents an ensemble of observations acquired by perturbing the observation vector y_t such that $y_t^i = y_t + \epsilon^i$. ϵ^i is a pertubation taken from a distribution with a bias and a standard deviation that represent the observation error. The subscript t refers to the time when a DA cycle is being carried out, which usually corresponds to the time appropriate observations are available. For the rest of this chapter, the subscript t will be withheld. The forecast-error covariance matrix is generated with the ensemble of background states:

$$P = \overline{(x^{f,i} - \overline{x^f})(x^{f,i} - \overline{x}^f)^T} \tag{2.2}$$

Where the overline denotes the average over the ensemble members. Minimizing J for each ensemble member yields the analysis ensemble:

$$x^{a,i} = x^{f,i} + PH^T(HPH^T + R)^{-1}(y^i - Hx^{f,i}) \tag{2.3}$$

with the Kalman gain $K = PH^T(HPH^T + R)^{-1}$. In all experiments exhibited in this study Equation (2.3) was used to estimate only the atmospheric state.

2.1.2 Parameter Estimation in DA

Primarily, DA techniques are used to estimate the state while keeping the model parameters fixed. Depending on how strongly correlated the atmospheric variables are to the parameters this can lead to error growth even if the initial state is known perfectly. A popular method to estimate model parameters during the DA cycle is *state augmentation* (Jazwinski, 1970). For this approach the parameter vector θ is appended to the state vector x, resulting in the *augmented state*:

$$\tilde{x}^{a/f,i} = \begin{bmatrix} x^{a/f,i} \\ \theta^{a/f,i} \end{bmatrix} \tag{2.4}$$

Where θ consists of the parameters that are wished to be estimated. To calculate the analysis for the augmented state one simply replaces $x^{a/f,i}$ with $\tilde{x}^{a/f,i}$ in Equations (2.2) and (2.3) and uses $\tilde{H}$ instead of H. Since the model parameters cannot be observed, $\tilde{H}$ is defined as:

$$\tilde{H}\tilde{x}^{a/f,i} = \tilde{H} \begin{bmatrix} x^{a/f,i} \\ \theta^{a/f,i} \end{bmatrix} = Hx^{a/f,i}. \tag{2.5}$$

This approach comes with additional challenges such as choosing a dynamical model and applying a localization scheme to the parameters to deal with underdispersion (Ruiz et al., 2013). In Ruckstuhl and Janjić (2018) these challenges were successfully approached by choosing stochastic dynamics to represent parameter uncertainty and introducing a global updating technique for localization in parameter space based on the spatial updating technique from Aksoy et al. (2006).

2.2 Machine Learning

Although the field of ML is vast and encompasses a wealth of different methods and algorithms, in this thesis we will concentrate on a subclass of ML algorithms - the artificial neural network (ANN). While the first steps towards ANNs occurred as early as in the middle of the 20th century (Mcculloch and Pitts, 1943), they only recently received a considerable amount of interest among all areas of scientific research with the advent of fast algorithms, parallel computing, and free machine learning libraries. ANNs are able to approximate complex, non-linear functions without any knowledge of the underlying function through a process called *Deep Learning* (LeCun et al., 2015).
But what is an ANN? Mathematically, an ANNs is itself a function $f(x)$, which maps an input $x \in \mathcal{R}^n$ to an output $y \in \mathcal{R}^m$. m, n are also called the input/output size respectively. $f(x)$ can be defined as a composition of other functions h_i such that:

$$f(x) = h_l(h_{l-1}(...h_2(h_1(x)))) \tag{2.6}$$

Each h_i is called one *layer* and its definition depends on the architecture (*functional model*) of the ANNs. In this study the non-linear weighted sum will be used such that:

$$f(x) = a_l(W_l a_{l-1}(W_{l-1}...a_2(W_2 a_1(W_1 x + b_1) + b_2)... + b_{l-1}) + b_l) \tag{2.7}$$

Where a_i is the non linear *activation function* of the i-th layer. Which activation function to use depends on the problem at hand. For this work the *Rectified Linear Unit (ReLU)* and the *Leaky Rectified Linear Unit (LeakyReLU)* were used, which are defined as:

$$ReLU(x) = max(0, x) \tag{2.8}$$

$$LeakyReLU(x) = max(0, x) + 0.01 * min(0, x) \tag{2.9}$$

$l - 1$ is the number of *hidden layers* and $\{W_i\} = W, \{b_i\} = b$ are sets of matrices/vectors with coefficients $W_i^{(k,l)}, b_i^{(k)}$ called the *weights/biases* of the ANN, which parameterize the ANN's functional model. For easier notation, it is possible to append the biases b onto the matrices W when simultaneously a 1 is appended onto x, which will be used from now on. Such an ANN is called *fully connected feed forward*. Since this notation is cumbersome, especially when the ANN becomes more complex, it is convenient to visualize it as a network structure where an arrow represents a single weight and a node depicts the weighted sum after the activation function has been applied.

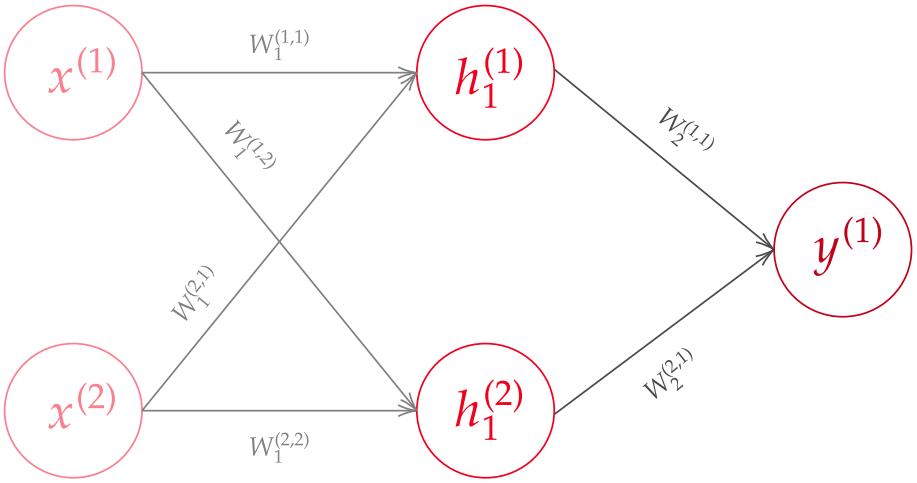

Figure 2.2: Artificial neural network visualized as network structure

In Figure 2.2 a simple ANN with $n = 2, m = 1$, 1 hidden layer and no biases is illustrated. Note that the size of the hidden layer, which in this example is 2, can be chosen freely. On its own the ANN is nothing special, it only becomes powerful when it is being *trained*. Because the initialization and the training of the weights depend on the type of ANN, further explanations can be found in the following subchapters. The one thing they have in common though is the fact that the ANN needs *training data* which is a set of input-output pairs $\{x_j\} = X, \{y_j\} = Y$ such that $f_{true}(x_j) = y_j$, where f_{true} represents the true function that is wished to be approximated. Note that it is possible to acquire X, Y without actually knowing f_{true}: training data for a *cats and dogs classifier (an ANN which takes pictures as input and predicts whether there was a cat or a dog in that picture)* can be produced by anybody who knows what a cat and a dog look like. f_{true} in this case is the inner mechanisms of the human brain and its biological neurons. This is also where the

name Artificial Neural Network was inspired from. Nevertheless, it should be noted that even one of the largest published ANN's with 160 billion parameters (Trask et al., 2015) is not even close to the complexity of the human brain with more than 100 trillion synapses (Nguyen, 2010).

2.2.1 Point Estimate Neural Networks

The name *point estimate neural network (NN)* was adopted from Jospin et al. (2020) and is used in this work to refer to a type of ANN which usually in literature is simply called *neural network*. This nomenclature was chosen to emphasize the distinction to Bayesian neural networks. For the rest of this study, we will refer to point estimate neural networks as NN and also use NN as a mathematical symbol to represent a forward pass through the network defined by its functional model. The training of the NN is achieved by approximating the minimal loss point W_{min} of a loss function $f_{loss}(NN(X, W), Y)$ over n_{tr} input-label pairs X, Y. *Stochastic gradient descent* is the most popular method of solving this high-dimensional optimization problem and can be summarized as:

1. Initialize numerical values W' for all coefficients of the weights either randomly or through prior knowledge
2. Calculate
$$f_{loss}(NN(X, W'), Y) = \frac{1}{n_{tr}} \sum_{j=1}^{n_{tr}} f_{loss}(NN(x_j, W'), y_j) \tag{2.10}$$
for each input-output pair of the training set by realizing a *forward pass* through the network
3. Update the weights by taking a step in the direction of the negative gradient
$$W' := W' - \frac{\eta}{n_{tr}} \sum_{j=1}^{n_{tr}} \nabla f_{loss}(NN(x_j, W'), y_j) \tag{2.11}$$
where η is called the *learning rate*
4. Go back to 2 until $W' \simeq W_{min}$.

In practice, the training set is usually split up into subsets called *mini-batches*. Once steps 3 and 4 have been applied on all mini-batches an *epoch* has been completed. In this study the *mean squared error (MSE)* was chosen as the loss function:

$$MSE(NN(X, W), Y) = \frac{1}{n_{tr}} \sum_{j=1}^{n_{tr}} (NN(x_j, W) - y_j)^2. \tag{2.12}$$

There have been many improvements successfully applied to the basic stochastic gradient descent algorithm. The one used in this work is the algorithm called *Adam*, which was proposed in Kingma and Ba (2014). Instead of having a constant learning rate, η is being adapted during the training individually for each weight based on the first and second moments of the gradients.

Once the training phase is finished, the current values for the weights W are being frozen and the NN can be used to predict an output for a given input x_{pred} by realizing a forward pass through the network:

$$NN(x_{pred}) = y_{pred}. \tag{2.13}$$

2.2.2 Bayesian Neural Networks

The definition of BNNs is not completely consistent across literature. In Jospin et al. (2020) it is defined as a type of ANN "...built by introducing stochastic components into the network..." and trained using *Bayesian inference* (MacKay, 1992). In Figure 2.3 the difference between stochastic and point estimate neural networks is visualized. Stochastic components can either be introduced as probability distributions over the activation functions or over the weights, although for this study the latter one was utilized as this is the more common one, i.e.

$$W \sim p(\omega). \tag{2.14}$$

We introduced a new variable ω here to emphasize the distinction between sampled weights W from their underlying probability distributions $p(\omega)$. After ascribing these *priors* $p(\omega)$, one can obtain the *likelihood of observations* $p(D|\omega)$ using some training data D. Note that we assume here that a functional model has already been chosen.

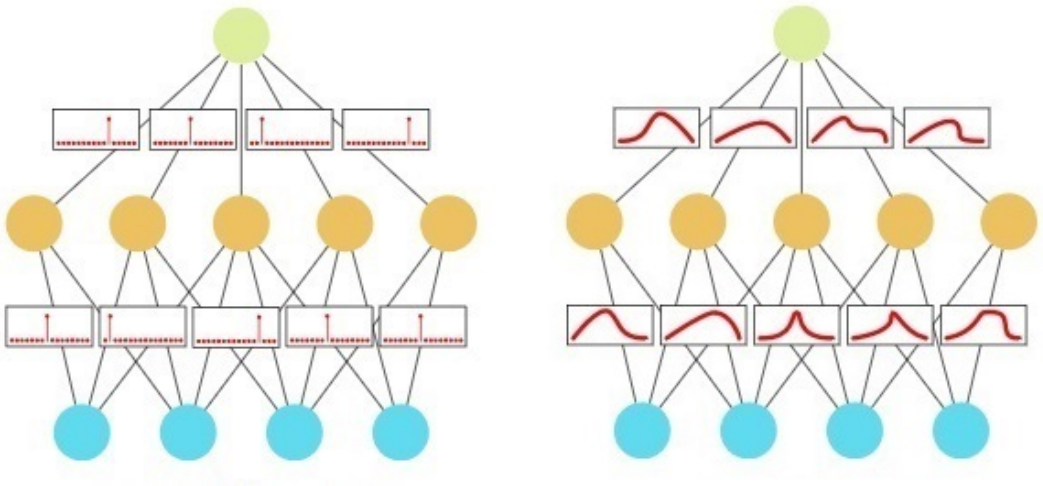

Figure 2.3: Distinction between point estimate neural networks (left) and stochastic neural networks (right) with probability distributions over the weights (Jospin et al., 2020)

The *evidence* $p(D)$ is then given by:

$$p(D) = \int_H p(D|\omega')p(\omega')d\omega'. \tag{2.15}$$

In the case of the NN we defined a "good" model as the one with weight coefficients that minimized a loss function. For the BNN a "good" model is one with

distributions that maximize the log evidence $log(p(D))$. The posterior distribution $p(\omega|D)$ is given by Bayes' rule:

$$p(\omega|D) = \frac{p(D|\omega)p(\omega)}{\int_\omega p(D|\omega')p(\omega')d\omega'} = \frac{p(D|\omega)p(\omega)}{p(D)} \tag{2.16}$$

So that given a new input x_{pred} the BNN's uncertainty can be obtained by:

$$p(y_{pred}|x_{pred}, D) = \int_\omega p(y_{pred}|x_{pred}, \omega')p(\omega'|D)d\omega'. \tag{2.17}$$

Although in practice it is usually sampled indirectly:

$$\begin{aligned} W &\sim p(\omega|D) \\ y_{pred} &= BNN_W(x_{pred}). \end{aligned} \tag{2.18}$$

Equation (2.18) can be used n_{sample} times to obtain a set of predictions $\{y^i_{pred}\}^{n_{sample}}_{i=1}$ from which mean and variance can be calculated to estimate the models predictive power and uncertainty. For the rest of this study we will use the notation $BNN(x_{pred}) = \{y^i_{pred}\}$ to summarize this process.
However, computing the evidence which is needed to get the posterior is usually intractable. Therefore in this study the variational inference approach is used to approximate the posterior $p(\omega|D)$ with a *variational distribution* $q_\phi(\omega)$, which is parameterized by the *distribution parameters* ϕ. During the training of the BNN the distribution parameters ϕ are learned such that $q_\phi(\omega)$ is as close to $p(\omega|D)$ as possible.
To measure the dissimilarity between two distributions $q_\phi(\omega)$ and $p(\omega|D)$ the Kullback-Leibler divergence (KL, Kullback and Leibler (1951)) is used:

$$KL(q||p) = \int_\omega q_\phi(\omega')log\left(\frac{q_\phi(\omega')}{p(\omega'|D)}\right)d\omega' = \mathbb{E}_q[log\, q_\phi(\omega)] - \mathbb{E}_q[log\, p(\omega|D)] \tag{2.19}$$

Although it seems like minimizing KL in this form is still not possible because the posterior appears directly in Equation (2.19), this is not the case. To show this another measure, the *evidence lower bound (ELBO)*, is introduced:

$$ELBO = \mathbb{E}_q[log\, p(\omega, D)] - \mathbb{E}_q[log\, q(\omega)]. \tag{2.20}$$

Where the joint probability $p(\omega, D) = p(D|\omega)p(\omega) = p(\omega|D)p(D)$ is used to easily show the relationship between ELBO and KL:

$$\begin{aligned} ELBO &= \mathbb{E}_q[log\, p(\omega, D)] - \mathbb{E}_q[log\, q_\phi(\omega)] \\ &= \mathbb{E}_q[log\, p(\omega|D)] + \mathbb{E}_q[log\, p(D)] - \mathbb{E}_q[log\, q_\phi(\omega)] \\ &= log(p(D)) - KL(q||p). \end{aligned} \tag{2.21}$$

Because $p(D)$ is independent of the posterior, maximizing $ELBO$ is equivalent to minimizing $KL(q||p)$. Additionally $KL(q||p) \geq 0$ as a result of Gibbs' inequality. Therefore ELBO is a lower bound of the log evidence:

$$ELBO \leq log(p(D)). \tag{2.22}$$

As a result, maximizing ELBO increases the log evidence and thereby making the BNN more accurate and moves the variational distribution $q_\phi(\omega)$ closer to the true posterior $p(\omega|D)$. The complexity of maximizing $ELBO$ depends on the choice of q_ϕ and usually, distributions from the exponential family are used to simplify training. For this study, Normal distributions were used to construct q_ϕ. A popular method to maximize $ELBO$ is stochastic variational inference. For a detailed derivation of this technique, we refer to Hoffman et al. (2013). Stochastic variational inference is stochastic gradient descent applied on variational inference and also carried out by iterating through all mini-batches of the available training data for several epochs. Again, once training has completed the values for the distribution parameters ϕ are frozen and the BNN can be used to make an ensemble of predictions by realizing several forward passes through the network with the same input x_{pred}, each time sampling W from $q_{\phi_{max}}$:

$$BNN(x_{pred}) = \{y_{pred,i}\}_{i=1}^{n_{sample}} \tag{2.23}$$

where the number of samples n_{sample} can be chosen freely. The choice of the prior distribution $p(\omega)$ and the family which the variational distribution q_ϕ is taken from is usually called the BNN's *stochastic model* to differentiate it from its functional model, which represents the number and type of hidden layers, size of hidden layers and activation functions.

2.2.3 Parameter Estimation in ML

Since parameter estimation using classical approaches is well-studied, we approach this challenge through the lens of machine learning. In Yadav et al. (2020) the coupling parameter of the two-level Lorenz-96 model (Lorenz, 2005) was estimated as a function of the resolved, large-scale state variable using a Gaussian Process (GP) (Rasmussen and Williams, 2006). The training data was generated by solving the Lorenz-96 model with different values of the coupling parameter and using snapshots from the time series of the state variable as the input. The parameter used to generate a respective time series served as the corresponding label. Additionally, the GP was compared to two types of deep neural networks and a simple linear regression where the GP outperformed the other methods in most of the experiments. The success was largely attributed to the true parameters being in fact jointly Gaussian. Further, it was noted that the GP decreases in accuracy considerably when the training data becomes noisy.

Chapter 3

Methods

In this chapter the specific methods used in this thesis are outlined in detail. The convective-scale model of the atmospheric state evolution from Würsch and Craig (2014) is summarized in Section 3.1. In Section 3.2 the parameter estimation algorithms are outlined and an overview of the architectures and training of the artificial neural networks is given. Finally, we briefly summarize the LRP algorithm used in this study in Section 3.3.

3.1 Modified Shallow Water Model

For this study the same numerical model, model parameters, and parameter bounds (Table 3.1) as in Ruckstuhl and Janjić (2018) were used to conduct the experiments but with ANNs estimating the parameters instead of the augmented state approach. Numerical *twin experiments* are a common approach to test new DA methods. Hereby the true state (*nature run*) of the atmosphere is generated by the physical model. Synthetic observations are produced by adding random perturbations to the true state. For all experiments in this study, such twin experiments were conducted using the *modified shallow water model* (Würsch and Craig, 2014). This model is computationally inexpensive but still represents the key space and time scales of storm developments. Thus it allows for easy testing of new methods for convective-scale DA.
The model is based on the shallow water equations for the fluid velocity u and the fluid height h with a modification of the geopotential ϕ to include conditional instability. Additionally, a variable for the rain r was added to mimic nature. The equations are as follows:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{\partial(\phi + c^2 r)}{\partial x} = \beta_u + D_u \frac{\partial^2 u}{\partial x^2} \tag{3.1}$$

$$\phi = \begin{cases} \phi_c & \text{if } h > h_c \\ gh & \text{else} \end{cases} \tag{3.2}$$

$$\frac{\partial r}{\partial t} + u\frac{\partial r}{\partial x} = D_r \frac{\partial^2 r}{\partial x^2} - \alpha r - \begin{cases} \delta \frac{\partial u}{\partial x} & \text{if } h > h_r \text{ and } \frac{\partial u}{\partial x} < 0 \\ 0 & \text{else} \end{cases} \tag{3.3}$$

$$\frac{\partial h}{\partial t} + \frac{\partial(uh)}{\partial x} = D_h \frac{\partial^2 h}{\partial x^2} \tag{3.4}$$

- D_u, D_r, D_h : diffusion constants
- $c^2 = g \times h_0$: gravity-wavespeed for absolute fluid layer $h_0 (h_0 < h_c)$
- δ : production rate of rain
- α : removal rate of rain

Convection is triggered by adding a low amplitude noise source β_u at random locations to the velocity at every model time step. When the fluid height h exceeds the threshold h_c, which represents the level of free convection, the geopotential is replaced by a lower constant value ϕ_c. The gradient of the geopotential forces fluid to the regions of lower geopotential, which then builds up the fluid height in those regions. Once h reaches the threshold h_r rain is being produced by adding rainwater mass to the geopotential. The removal of rain is mimicked by a linear relaxation towards zero. For the experimental set-up a one dimensional grid of length 125 km with 250 grid points was used, which yields a state vector of the form:

$$x = \begin{bmatrix} u \\ h \\ r \end{bmatrix} \in \mathbb{R}^{750}. \tag{3.5}$$

The model parameters which were chosen to be estimated are the rain removal rate α, the low constant value for the geopotential ϕ_c and the threshold for the fluid height h_r. Assuming all of the model parameters are constant in space the resulting parameter vector is:

$$\theta = \begin{bmatrix} \alpha \\ \phi_c \\ h_r \end{bmatrix} \in \mathbb{R}^3. \tag{3.6}$$

Observations are generated from the nature run every 60 model time steps by adding a Gaussian error to u and h and a lognormal error to r to keep its positivity. To simulate radar data only the grid points where $r > 0.005$ were observed. Furthermore, wind observations of 25% of the remaining grid points were added. The model parameters for the nature run are taken from uniform distributions. The upper and lower bounds of the uniform distributions for the model parameters as well as biases and standard deviations of the observational errors are summarized in Table 3.1 and Table 3.2 respectively.

Parameter	Lower Bound	Upper Bound
α	0.0003	0.001
ϕ_c	899.7	899.9
h_r	90.15	90.25

Table 3.1: Lower and upper bounds for the uniform distributions of the model parameters

Variable	Mean	Standard Deviation
u	0	0.001
h	0	0.02
r	0.001	1e-7

Table 3.2: Means and standard deviations for the normal distributions of the observational errors

Functional Model		
Type of Layer	Size (input x output)	Activation Function
Linear	750 x 31	ReLU
Batch-Norm	31 x 31	None
Dropout (p=0.5)	31 x 31	None
Linear	31 x 19	ReLU
Linear	19 x 11	ReLU
Linear	11 x 3	Sigmoid (only at evaluation)
Training		
Optimizer	Adam	
Mini-Batch-Size	32	
Number of Epochs	150	

Table 3.3: Functional model and training specifics of *NN*

3.2 Parameter Estimation Algorithms

Deep learning for model parameter estimation is not well developed. The methods described in this section can be seen as the results of a first exploration within this field. Two types of artificial neural networks - a deep ensemble of point estimate neural networks (Section 3.2.1) and a Bayesian neural network (Section 3.2.2) - were trained on the background state produced by the modified shallow water model. The goal is to estimate the global model parameters $\alpha, \phi_c\, h_r$ as a function of the state consisting of the observable atmospheric variables u, h, r.

3.2.1 Deep Ensemble of Point Estimate Neural Networks

Functional Model

A natural choice for the output of the NN is the model parameter vector $\theta = \left[\begin{smallmatrix}\alpha\\ \phi_c\\ h_c\end{smallmatrix}\right] \in R^3$ since these are the unknown variables to be estimated. The input is generally determined by the available data, although somewhat arbitrary. Developing a theoretical basis for selecting appropriate features as inputs is an active area of research (Varma, 2020). For the offline approach all three atmospheric variables of the whole grid $x = \left[\begin{smallmatrix}u\\ h\\ r\end{smallmatrix}\right] \in R^{750}$ at one point in time were chosen as the input, yielding an input size of 750 for

$$NN(x) = \theta. \tag{3.7}$$

In Krasnopolsky et al. (2013) a multi-output regression problem with data simulated by a cloud-resolving model was successfully approached using an ensemble of fully connected feed-forward NNs. Therefore this basic architecture was adopted with an additional batch normalization layer to accelerate the training (Ioffe and Szegedy, 2015) and a dropout layer to increase accuracy and minimize overfitting (Labach et al., 2019). A summary of the architecture can be found in Table 3.3.

Training

To generate the input-output pairs for the training, first $n_{tr} = 100.000$ sets of parameters were taken randomly from the uniform distributions specified in Table 3.1:

$$\theta^j = \begin{bmatrix}\alpha^j\\ \phi_c^j\\ h_c^j\end{bmatrix} \sim \begin{bmatrix}U(l_\alpha, u_\alpha)\\ U(l_\phi, u_\phi)\\ U(l_h, u_h)\end{bmatrix}. \tag{3.8}$$

For each set of parameters θ^j the shallow water model was initialized with the same starting conditions and run for $t_0 = 1000$ model time steps resulting in:

$$M(\theta^j, x^{init}) = \begin{bmatrix} u^j_{t_0} \\ h^j_{t_0} \\ r^j_{t_0} \end{bmatrix}. \tag{3.9}$$

Where j goes from $1...n_{tr}$, M denotes the modified shallow water model and x^{init} is the same initial state vector for all n_{tr} input vectors. Note that only the state vector at $t_0 = 1000$ was used for the training. We chose 1000 here because this corresponds to the point in time when the first observations were available for the DA. Training the NN on state vectors taken from a wide range in time results in poorer performance. This is caused by the NN not being able to distinguish between differences in the states due to different parameters versus due to the time evolution of the state. Additionally, the input data was augmented during the training by adding pertubations taken from distributions with means and standard deviations corresponding to the values specified in Table 3.2. For each input sample at $t = 1000$ 3 perturbed samples were added during the training resulting in a training size of 400.000. This is a common technique to increase the training size and enhance the generalization capabilities of a NN (Rusak et al., 2020).

Uncertainty Estimation

To quantify the uncertainty of the parameter estimation the method of deep ensembles from Lakshminarayanan et al. (2017) was adopted. This is an easy implementable approach, where an ensemble of neural networks $\{NN_k\}_{k=1}^{n_{NN}}$ consisting of n_{NN} members with the same functional model but random initial weights are trained independently. During the DA cycles, each analysis ensemble member $x^{a,i}_t$ was used as input for each NN ensemble member resulting in $\{\theta^{i,k}_t\} \rightarrow n_{NN} * N_{ens}$ parameter estimates. To obtain N_{ens} parameter vectors out of this distribution two methods were implemented and compared:

$$\frac{1}{n_{NN}} \sum_{k=1}^{n_{NN}} NN^k(x^{a,i}_t) = \theta^i_t \tag{3.10}$$

$$\theta^i_t \sim Beta(\mu_t, var_t) \tag{3.11}$$

μ_t and var_t are the mean and variance from $\{\theta^{i,k}_t\}$. A beta distribution was chosen here to keep the parameters bounded. From here on we will refer to (3.10) as the *mean* method and to (3.11) as the *beta* method. To investigate the influence of the *NN ensemble size* (n_{NN}), experiments for $n_{NN} = 5, 10, 15$ were conducted.

Combining with Data Asimilation

After the training was completed, the ensemble of NNs was used during the DA cycle to estimate parameters. The estimated state vector generated by the stochastic EnKF was used as input (Figure 3.1). In the following the algorithm is summarized:

1. DA cycle < 0: Training

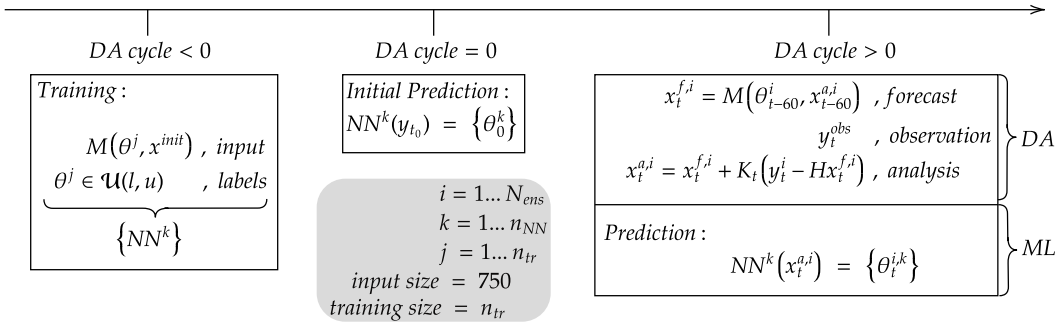

Figure 3.1: Scheme of the integration of the deep ensemble of point estimate neural networks into the data assimilation cycle

2. DA cycle = 0: To initialize the background state for the DA the observations were used as input for the NNs (if state was only partially observed, observations were interpolated first)

3. DA cycle > 0: Whenever observations were available (every 60 model time steps)

   (i) an analysis $x_t^{a,i}$ was calculated using the stochastic EnKF and a forecast-error covariance localization (Gaspari and Cohn, 1999) of 6 grid points

   (ii) for each analysis ensemble member $x_t^{a,i}$ an ensemble of parameters $\{\theta_t^{i,k}\}$ was estimated using the NN ensemble

   (iii) N_{ens} parameter vectors were calculated either according to the mean or to the beta method and used as parameters of the modified shallow water model for the short forecast until the next observations were available

3.2.2 Bayesian Neural Networks

Bayesian neural networks are known to be data efficient, less prone to overfitting, and provide a natural way of quantifying uncertainty (Jospin et al., 2020) without the need to train a whole ensemble of NNs. The objective of the methods described in the following section is twofold. First, to compare the accuracy and uncertainty of a Bayesian neural network (BNN_0) with a deep ensemble of point estimate neural networks. Second, to investigate the feasibility of training BNNs every time new observations are available (BNN_t) using a realistic number of forecast ensemble members as the training data.

Functional Model

In this study 2 Bayesian neural networks, BNN_0 and BNN_t (Tables 3.4 and 3.5), were trained. Similar to the point estimate neural network, fully connected feed-forward models with the atmospheric state as the input and the parameters as the output were chosen as the functional model:

$$BNN(x) = \theta. \tag{3.12}$$

The hyperparameters such as neurons per hidden layer, position of batch normalization layer, and number of epochs were optimized independently and therefore slightly differ from the functional model of the point estimate neural network (Table 3.3).

Functional Model			Stochastic Model	
Type of Layer	Size (input x output)	Activation Function	Priors $p(W_i^{(k,l)})$	$\mathcal{N}(0,1)$
Batch-Norm	750 x 750	None	Variational distributions $q_\phi(W_i^{(k,l)})$	$\mathcal{N}(\mu,\sigma)$
Linear	750 x 20	LeakyReLU	Training	
Linear	20 x 20	LeakyReLU	Optimizer	Adam
Linear	20 x 20	LeakyReLU	Mini-Batch Size	32
Linear	20 x 3	Sigmoid (only at evaluation)	Number of Epochs	3

Table 3.4: Functional model, stochastic model and training specifics of BNN_0

Functional Model			Stochastic Model	
Type of Layer	Size (input x output)	Activation Function	Priors $p(W_i^{(k,l)})$	$\mathcal{N}(0,1)$
Linear	$min\{dim(\tilde{y}^{obs}), 124\}$ x 2	LeakyReLU	Variational distributions $q_\phi(W_i^{(k,l)})$	$\mathcal{N}(\mu,\sigma)$
Linear	2 x 2	LeakyReLU	Training	
Batch-Norm	2 x 2	None	Optimizer	Adam
Linear	2 x 2	LeakyReLU	Mini-Batch Size	32
Linear	2 x 3	Sigmoid (only at evaluation)	Number of Epochs	9

Table 3.5: Functional model, stochastic model and training specifics of BNN_t

Training

BNN_0 was trained on the same training data that was used for the point estimate neural network: 100.000 snapshots in time of the atsmopheric state at $t = 1000$. BNN_0 has in total about 33.000 trainable weights. This large number is no problem if there is enough training data available.
On the other hand, BNN_t is trained on snapshots of the atmospheric forecast state for $t > 1000$ during the data assimilation cycle whenever new observations are available. Since the number of analysis/forecast ensemble members in a real-life scenario is restricted by computational limitations, the training data size is much smaller than that used for BNN_0. Therefore it is necessary to reduce the number of trainable weights for BNN_t. The first modification is to reduce the input size. Experiments from Ruckstuhl and Janjić (2018) indicate that the rain r and fluid height h are stronger correlated to the parameters than the wind u. Hence instead of using all 3 atmospheric variables as the input, only r and h were used. Since BNN_t is trained from scratch each time, the input size can be left variable. This allows to only train on those grid points that are actually observed. Additionally, because we assume the true parameters to be constant over the whole grid, it might not be necessary to use all observed grid points as input. Therefore if more than 62 gridpoints (about 25 % of the whole grid) were observed, only those 62 grid points with the highest observed values for r were used. $\tilde{x}^{f,i}$ in Figure 3.2 refers to this reduced background state. To further reduce the number of learnable weights, the number of neurons per hidden layer was decreased from 20 to 2. In total, this results in a maximum of around 540 learnable weights for BNN_t. The resulting input for the training of BNN_t is then given by

$$H_t\tilde{x}_\tau^{f,i} + \epsilon^i \tag{3.13}$$

with the observation operator H_t and $i = 1, ..., N_{ens}$. The 10 previous points in time $\tau = t - 9, ..., t$ were used to increase the training size from N_{ens} to $N_{ens} \cdot 10$. The labels for these inputs are simply the model parameters θ^i_{t-60} from the previous estimation. Noise corresponding to the observational error specified in Table 3.2 was added during the training for the same reasons as for the offline training. Details of the architecture for BNN_t can be found in Table 3.5.
For BNN_t 9 training epochs were necessary to reach a minimum in the validation loss, which is most likely caused by the reduced training size compared to the offline training.

Uncertainty Estimation

A widely used default for the priors of BNNs are normal distributions with mean 0 and standard deviation σ (Jospin et al., 2020). After evaluating the trained weights of the point estimate neural networks it seemed appropriate to set $\sigma = 1$. When using BNNs to estimate parameters, the number of outputs is not restricted. By performing several forward passes through the trained network with the same input a distribution of outputs can be generated.

Combining with Data Assimilation

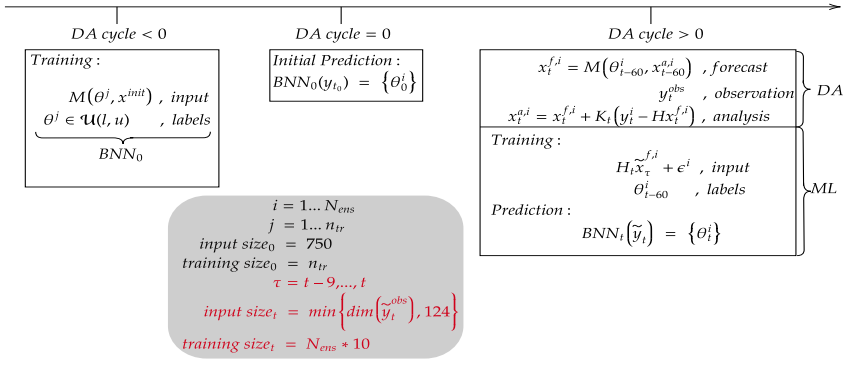

Figure 3.2: Scheme of the integration of the Bayesian neural networks into the data assimilation cycle

The combination of the Bayesian neural network with the data assimilation is similar to that of the point estimate neural network. In the following the algorithm is summarized:

1. DA cycle < 0: Training of BNN_0
2. DA cycle $= 0$: To initialize the background state for the DA the observations were used as input for BNN_0 (if state was only partially observed, observations were interpolated first)
3. DA cycle > 0: Whenever observations were available (every 60 model time steps)
   (i) an analysis $x_t^{a,i}$ was calculated using the stochastic EnKF and a forecast-error covariance localization (Gaspari and Cohn, 1999) of 6 grid points

(ii) a new Bayesian neural network, BNN_t, was trained on the transformed forecast ensemble computed by $H_t\tilde{x}_\tau^{f,i} + \epsilon^i$

(iii) the observation vector $\tilde{y}_t$ was passed through the trained network BNN_t N_{ens} times to generate an ensemble of parameter estimates

(iv) the N_{ens} parameter vectors were used as parameters of the modified shallow water model for the short forecast until the next observations were available

3.3 How Do Machines Learn?

To obtain more insight into how the ANNs trained in this work are learning to estimate the model parameters from the observable state, the method of *Layer-Wise Relevance Propagation* was utilized. LRP is a visualization tool, which takes a trained ANN and an ANN input sample, which can be either from the training or test datasets, as the input and produces an *LRP heatmap* as the output. The LRP heatmap is a vector of the same size as the ANN input and those entries with higher numerical values can be interpreted as being more relevant for the ANN's prediction as the ones with lower values (Figure 3.3).

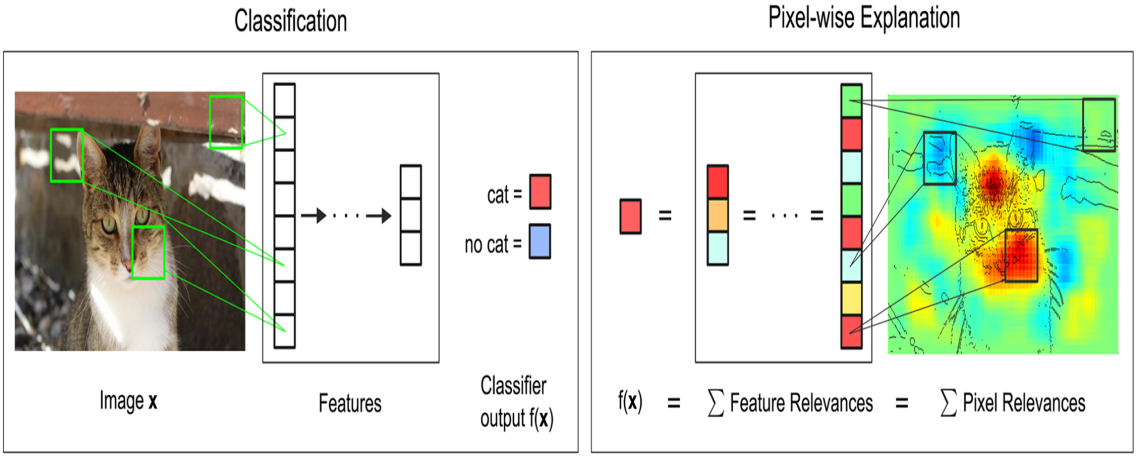

Figure 3.3: Visualization of the LRP algorithm applied on an image classification ANN (Bach et al., 2015)

This method has been introduced first to the field of computer vision by Bach et al. (2015) and the *PyTorch* implementation used in this study is from Böhle et al. (2019). For an in-depth explanation of the algorithm, we refer to Toms et al. (2020), who recently introduced LRP to the geosciences. The underlying idea of LRP is to calculate a *relevance* for each input pixel by taking a specific ANN output, which in this case would be either α, ϕ_c or h_r, and propagating it back through the network according to a certain set of propagation rules. Applying LRP to the ANNs trained in this study could thus give insight into which grid points and atmospheric variables were most relevant for each of the three parameters.

Chapter 4

Results

The experiments illustrated in this chapter investigate different techniques to incorporate the ML generated parameter estimates into the DA cycle and their influence on the initial state errors. The sensitivity of the state ensemble size, the observation coverage, and the neural network size are examined. Finally, LRP is utilized to investigate the influence of the model parameters to the model states by visualizing which grid points and atmospheric variables are most relevant for the ANNs prediction.

4.1 Metrics

The metrics used to display the experimental results for a variable λ are the *Root Mean Squared Error (RMSE)* and the *spread* of the estimated ensemble. Given the true state of a variable $\lambda^{tr} \in \mathbb{R}^n$, the ith member of the estimated ensemble $\lambda^i \in \mathbb{R}^n$ and its corresponding mean $\overline{\lambda} = \frac{1}{N_{ens}} \sum_{i=1}^{N_{ens}} \lambda^i$, the metrics are defined as:

$$RMSE(\lambda) = \sqrt{\frac{1}{n} \sum_{j=1}^{n} (\lambda^{tr} - \overline{\lambda})_j^2} \tag{4.1}$$

$$spread(\lambda) = \sqrt{\frac{1}{N_{ens} - 1} \sum_{i=1}^{N_{ens}} (\lambda^i - \overline{\lambda})^2}. \tag{4.2}$$

The RMSE is a measure of the accuracy of an estimator, where a lower value means that the estimator fits the data better. The spread is a measure of the uncertainty of the estimated ensemble. Therefore a larger value of the RMSE should correspond to a larger value in the spread. For each experiment shown in this chapter 100 simulations were conducted, where for each of these the parameters of the nature run were taken randomly from the bounded uniform distributions specified in Table 3.1. The metrics were averaged over all simulations to obtain statistically significant values. Additionally, the spread was averaged over all grid points for easier comparison. Finally, to evaluate the performance of the NN and of the BNN the *coefficient of determination (R^2)* will be utilized in Section 4.2. Given a set of N parameter predictions θ^i, their corresponding ground truths θ^i_{tr}, and the

ground truth's mean $\overline{\theta}_{tr} = \frac{1}{N}\sum_{i=1}^{N}\theta_{tr}^{i}$ MAE and R^2 are defined as:

$$R^2(\theta, \theta_{tr}) = 1 - \frac{\sum_{i=1}^{N}(\theta_{tr}^{i} - \overline{\theta}_{tr})^2}{\sum_{i=1}^{N}(\theta_{tr}^{i} - \theta^{i})^2}. \tag{4.3}$$

For a perfect model that always predicts the same value as the ground truth $R^2 = 1$. A model which always predicts the average of the ground truth $\overline{\theta}_{tr}$ would yield $R^2 = 0$. A model with $R^2 < 0$ performs worse than simply predicting the average of the ground truth for all inputs.

4.2 Performance

For the first comparison between point estimate neural networks and Bayesian neural networks 500 samples from the test dataset, which the ANNs were not trained on, were used to estimate the parameters. Each output ensemble was averaged and then plotted against its corresponding ground truth. Additionally, to rank the results obtained by the ANNs a simple linear regression (LR) model was fitted to the same training data (Figure 4.1). As a benchmark, the ideal output was plotted as well, which corresponds to the black lines with slope 1. For a fair comparison, the best performing ANNs of each type were used in these experiments.

model	α	ϕ_c	h_r
NN	0.53	0.44	0.62
BNN	0.79	0.74	0.75
LR	0.41	0.26	-0.52

Table 4.1: R^2 of the parameter predictions plotted in Figure 4.1 for NN, BNN and LR

The BNN outperforms the NN as well as the LR in all three parameters while the LR has the lowest R^2 scores for all parameters (Table 4.1). While the BNN has similar R^2 scores for the different parameters the NN's and LR's performance varies greatly between them. For h_r the LR performs even worse than a benchline model which would predict the average value of the bounds for all inputs. The scatter plot in Figure 4.1 emphasizes that both ANNs are slightly overestimating low parameter values while underestimating high parameter values. in correspondence with the R^2 scores in Table 4.1 the NN (blue) seems to have problems correctly predicting especially ϕ_c and low values for all parameters while the BNN (red) displays a rather uniform predictive power for all three parameters. The simple LR (green) predicts values that are greatly out of bounds for all parameters.

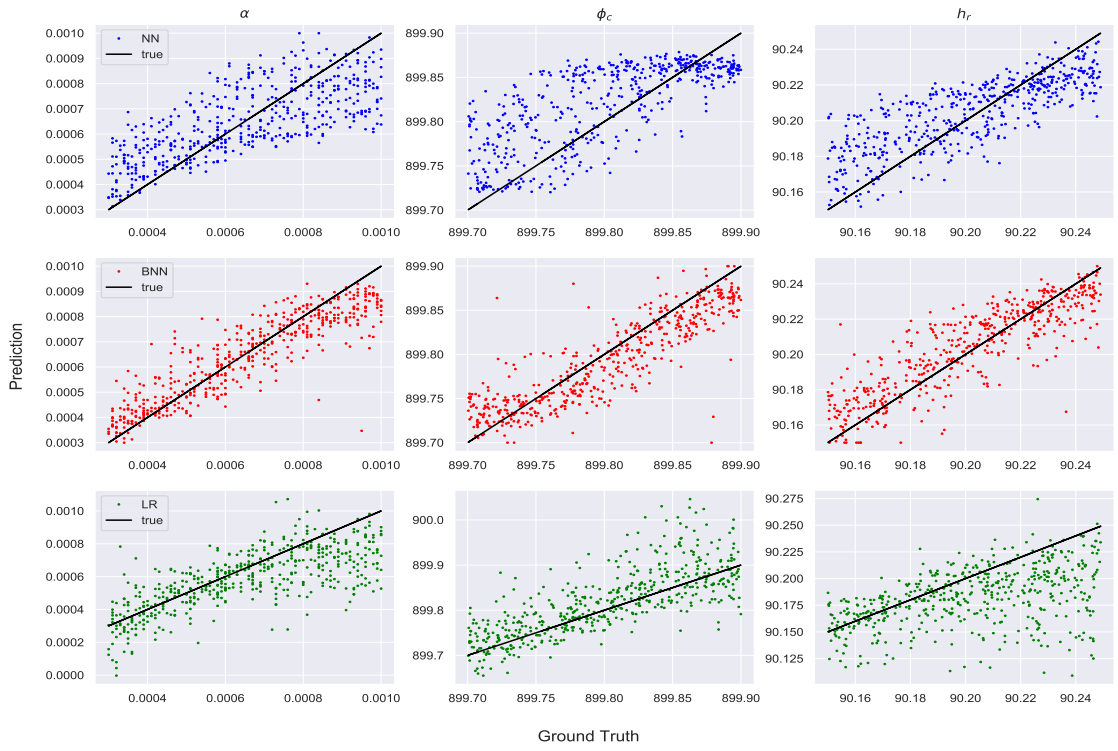

Figure 4.1: Output of best performing NN (blue dots), BNN (red dots), and simple LR (green dots) against ground truths and ideal output (black lines) of 500 samples

4.3 Time Evolution

Since the ANNs are trained on only one point in time, the question arises how they perform when using states from later points in time as the input and comparing this with BNN_t, which is constantly retrained in time. If the predictive power does not significantly decrease it would only be necessary to train the ANN once and it could then be used to predict parameters whenever necessary, even for changing parameters. To investigate this, time evolution experiments were conducted for all ANNs described in section 3.2 and compared with each other in the following subsections.

4.3.1 Point Estimate Neural Networks

For the time evolution experiments the first observations were taken from the nature run after a spinup time of 1000 model time steps and used to estimate the initial parameters:

$$NN^k(y_{t_0}^{obs}) = \{\theta_0^k\} \tag{4.4}$$

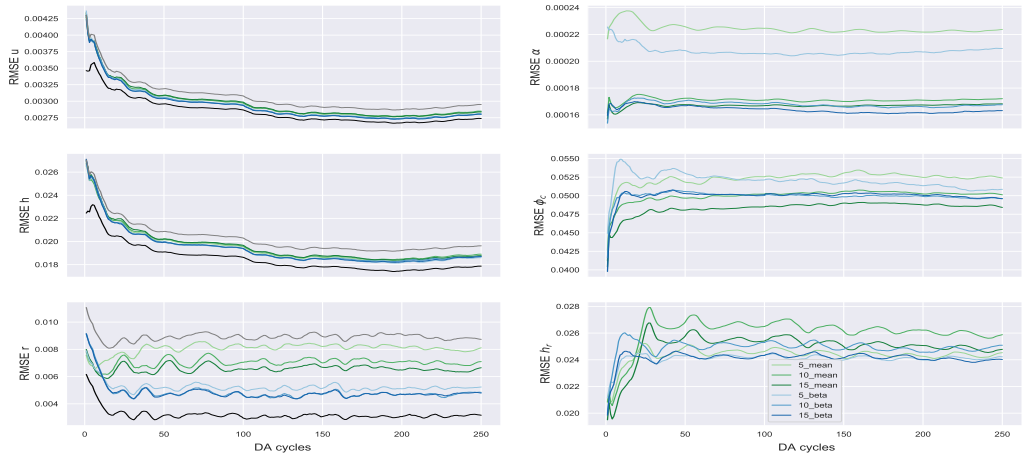

Figure 4.2: Time evolution of averaged RMSEs of atmospheric variable estimates (left) and parameter estimates (right) with 50 analysis ensemble members and for comparison, true parameters were known exactly and used for the background states (black) and parameters were not known and not estimated (gray)

For the mean method $\overline{\theta}_0 = \frac{1}{n_{NN}} \sum_{k=1}^{n_{NN}} \theta_0^k$ was used to initialize the background states of all state ensemble members. For the beta method $\overline{\theta}_0$ and the variance of $\{\theta_0^k\}$ were used to fit a beta distribution to the initial estimation. For all background ensemble members, the parameters were taken from this beta distribution. The background state was also run for 1000 model time steps before starting with the DA cycles. From then on, the NN was used as described in subsection 3.2.1. For Figures 4.2 and 4.3 the RMSEs were plotted against time in DA cycles. Note that between two points on the x-axis lie 60 model time steps. For all experiments, 50 ensemble members for the analysis and background were used. The RMSE of the parameters is smallest at the beginning of the time evolution because DA cycle = 0 corresponds to 1000 model time steps. After that, the RMSEs grow for about 50 cycles and then oscillate around a relatively constant value with a slightly negative slope (Figure 4.2, right). Generally, the error in the parameters and atmospheric variables decreases with bigger NN ensemble sizes (Figure 4.4) while their ensemble spread slightly decreases (Figure 4.5). The beta method outperforms the mean method in all experiments and is therefore used for all further experiments. The beta method is not only more accurate but also manages to increase the spread of all atmospheric variables (Figure 4.3, left). The influence of the parameter estimation is strongest for the rain r. This is due to the strong correlation of the parameters with the rain, especially the rain threshold h_r and the rain removal rate α.

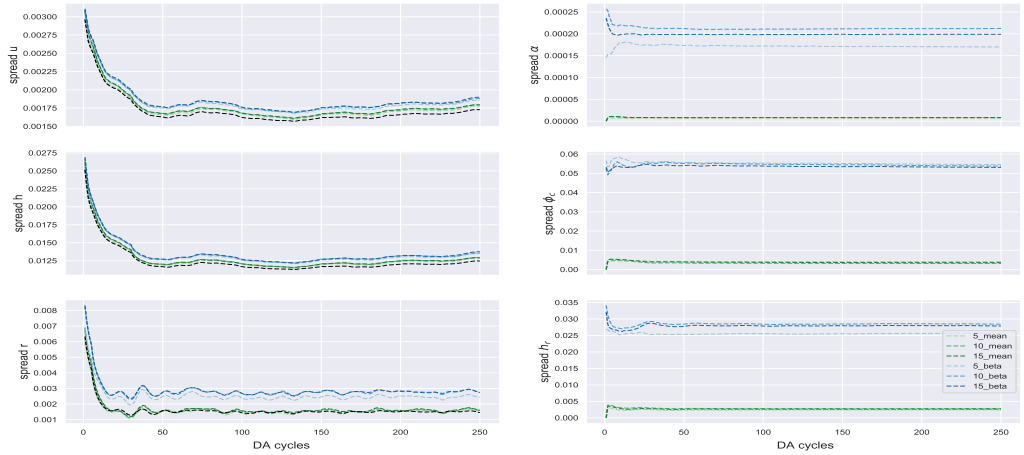

Figure 4.3: Time evolution of averaged ensemble spread of atmospheric variable estimates (left) and parameter estimates (right) with 50 analysis ensemble members and for comparison, true parameters were known exactly and used for the background states (black) and parameters were not known and not estimated (gray)

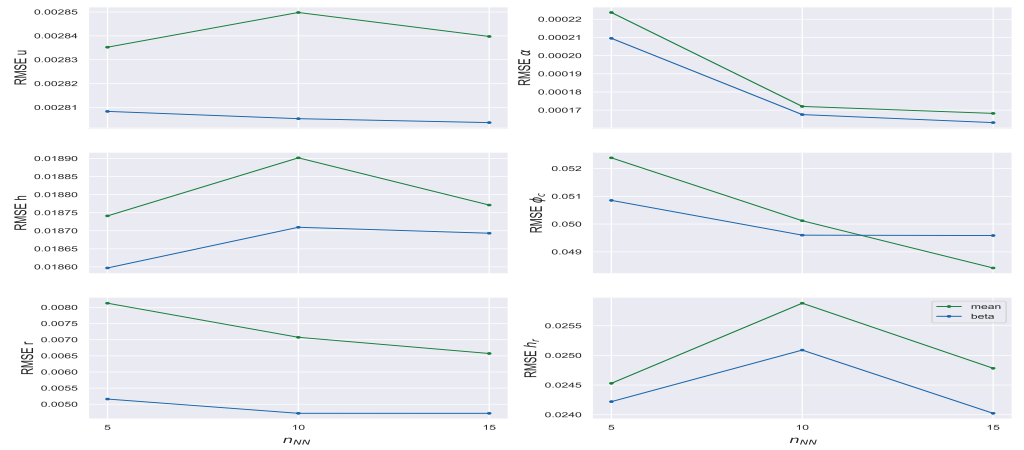

Figure 4.4: Data from Figure 4.2 averaged over last 100 DA cycles

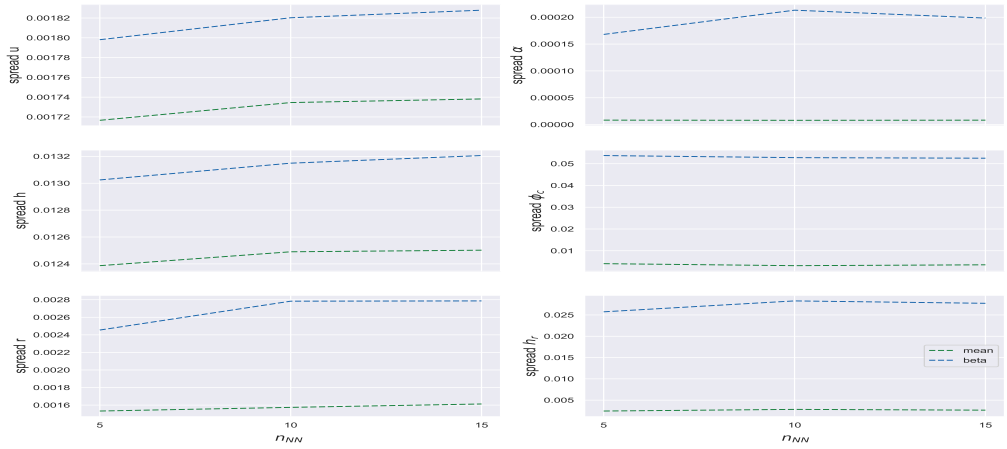

Figure 4.5: Data from Figure 4.3 averaged over last 100 DA cycles

4.3.2 Bayesian Neural Network

The same time evolution experiments were conducted for the BNN algorithm described in section 3.2.2. In Figures 4.6 and 4.7 this is referred to as $BNN_0 + BNN_t$. For comparison, the case where only BNN_0 is used throughout all 250 DA cycles and the beta method with 15 NN ensemble members were plotted as well. The errors for BNN_0 and 15_*beta* are directly comparable as they were both trained on the same training data and the same inputs were used for the parameter predictions. For the initial estimate at DA cycle = 0 BNN_0 outperforms the NN in all experiments. The error in α was reduced by 50 %, in ϕ_c by 38 % and in h_r by 40 %. Over time, however, the RMSEs of the parameters increase a lot faster for BNN_0 compared to 15_*beta*. The RMSEs of the atmospheric variables converge for both methods over time. $BNN_0 + BNN_t$ also increases the RMSEs over time, but to a lesser extent compared to the other methods. The increase in the RMSEs was likely caused by several factors. On one hand, the training data for BNN_0 was generated using the same initial state for all sets of parameters. For BNN_t each sample of the training data was generated by using a different set of parameters for each ensemble member of the analysis. Therefore the differences in the training data are not only due to different parameters but also due to different initial states. This could make it harder for the BNN to find distinct characteristics for the parameter estimation. On the other hand, the much smaller training size of 500 (10 points in time for each of the 50 background ensemble members) used in these experiments compared to the 100.000 used for BNN_0. Although the latter could be improved by simply using more forecast ensemble members.

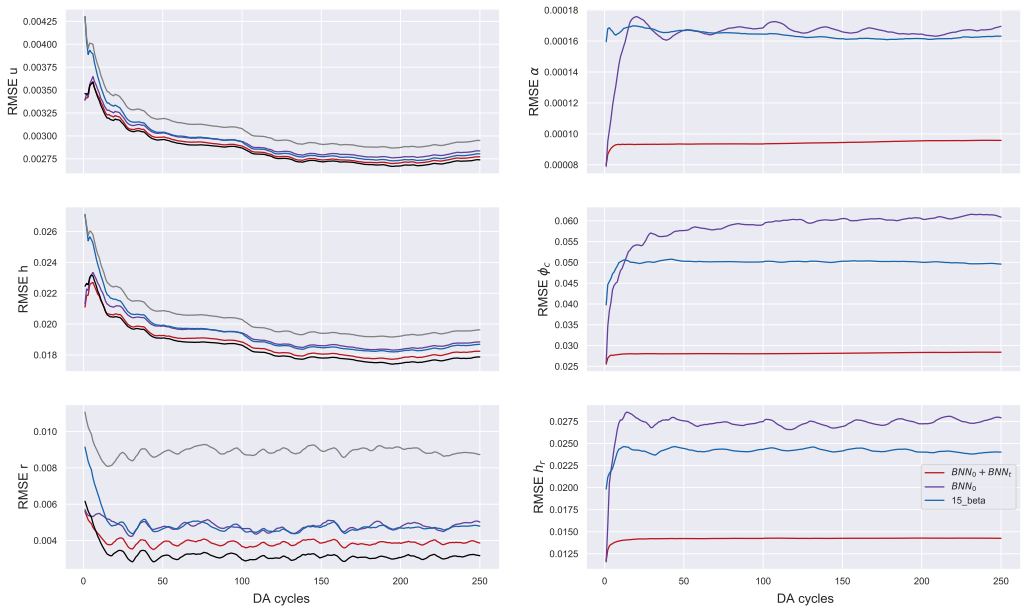

Figure 4.6: Time evolution of averaged RMSEs of atmospheric variable estimates (left) and parameter estimates (right) with 50 analysis ensemble members and for comparison, true parameters were known exactly and used for the background states (black) and parameters were not known and not estimated (gray)

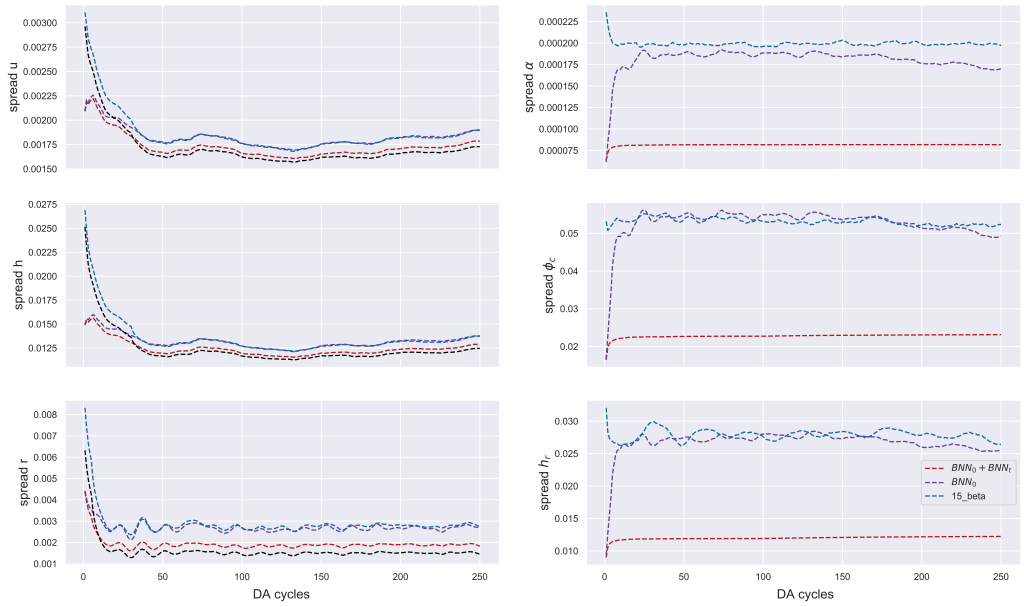

Figure 4.7: Time evolution of averaged ensemble spread of atmospheric variable estimates (left) and parameter estimates (right) with 50 analysis ensemble members and for comparison, true parameters were known exactly and used for the background states (black) and parameters were not known and not estimated (gray)

4.4 Distribution of Parameter Estimates

Histograms of the parameter estimates for one single experiment were plotted for the mean and beta method with 15 NN ensemble members, as well as for both BNNs. Since this is only a single experiment, the results shown here are not statistically significant. However, they still illustrate the key differences between the methods. To also investigate the change of the distributions over time, a histogram for each method was plotted at DA cycle = 1 (Figures 4.8 to 4.10, upper panel) and at DA cycle = 250 (Figures 4.8 to 4.10, lower panel). As a reminder, the mean method (green) takes the mean over all NN ensemble members for each analysis ensemble member. Therefore it does not actually portray the uncertainty of the parameter estimation of the NN, but rather the spread caused by the analysis. The fact that this spread shrinks over time is therefore in accordance with the results from Figures 4.3 and 4.7. The beta method (blue) on the other hand embodies much more the true distribution of the NN estimate, which reaches over the whole range with accumulations close to the true value. For this particular experiment, the prediction of the beta method actually improves over time. This can also be observed in Figure 4.2, where the RMSEs reach a maximum between DA cycles 0 and 50 and then slowly decrease over time. The initial estimate of BNN_0 is close to a Gaussian distribution with the mean near the true value and a small variance. Nonetheless, over time the distribution spreads out over the whole range with aggregations near the edges. This can be explained by the sigmoid activation function, that was used during evaluation: if a prediction was outside the given range, it was mapped to the bounds. It can therefore be concluded that the true spread of the BNN_0 is much larger. This is expected behavior from BNNs that is used on input, which differs a lot from its training data. The initial distribution of BNN_t at DA cycle = 1 also seems to be Gaussian and is similar to that of BNN_0. After some time at DA cycle = 250, most of the estimates of BNN_t lie very close to the true value with some outliers. These outliers could be the reason for the increase in the RMSE and spread over time in Figures 4.6 and 4.7 (right).

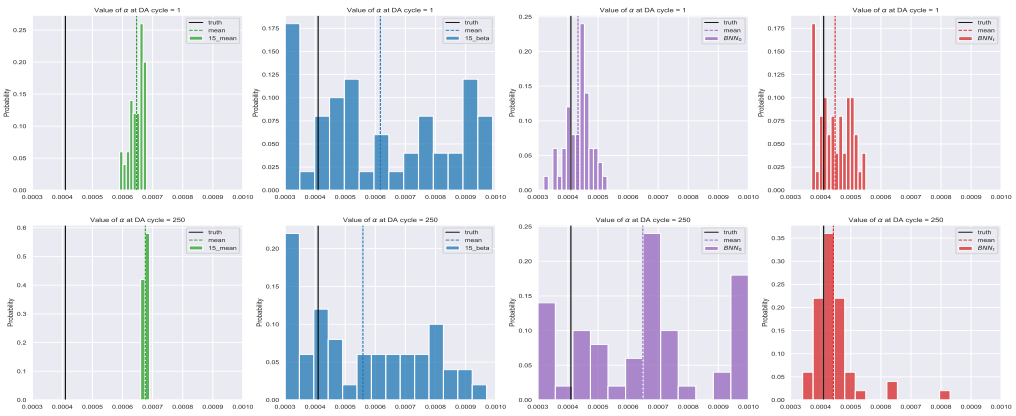

Figure 4.8: Histogramms of estimates for rain removal rate α for one single experiment at DA cycle = 1 (upper panel) and at DA cycle = 250 (lower panel)

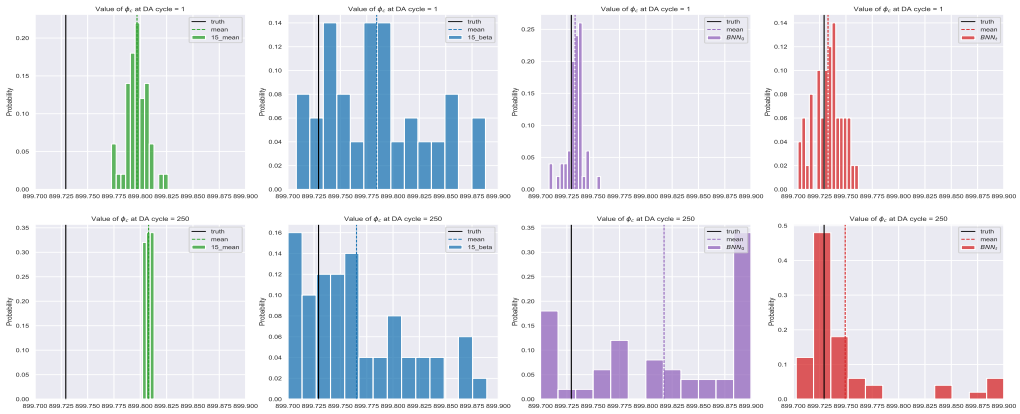

Figure 4.9: Histogramms of estimates for constant geopotential ϕ_c for one single experiment at DA cycle = 1 (upper panel) and at DA cycle = 250 (lower panel)

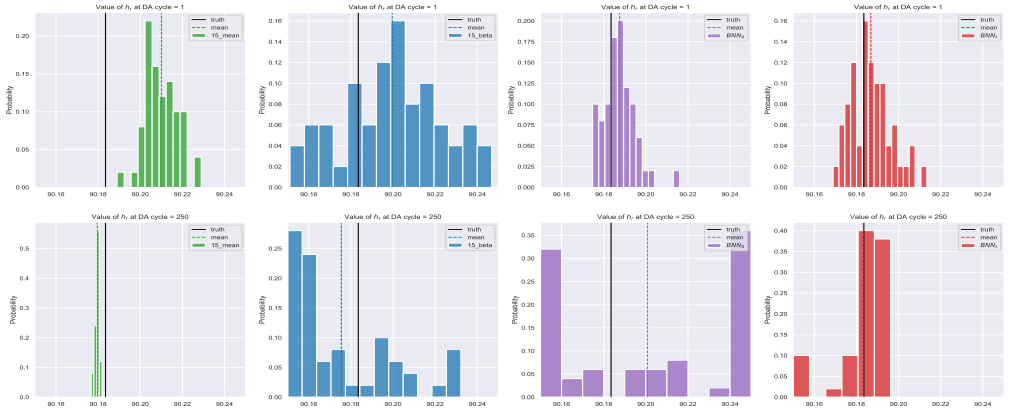

Figure 4.10: Histogramms of estimates for rain threshold h_r for one single experiment at DA cycle = 1 (upper panel) and at DA cycle = 250 (lower panel)

4.5 Sensitivity Experiments

The sensitivity of the analysis RMSE and analysis ensemble spread to the number of state ensemble members and observation coverage was studied for different setups to compare the performance and statistics of the different methods with the best and worst case szenarios and to investigate the capabilities of the ANNs under sparse conditions. Additionally, the sensitivity to the BNN size, here defined as neurons per hidden layer, was examined. For the sensitivity experiments illustrated in the following section the same experiments as described in section 4.3 were conducted and averaged over the last 100 DA cycles.

4.5.1 Setups

In total, 6 setups were studied:

1. *true* (black): the true values of the parameters were known and used for the background state throughout all 250 DA cycles
2. *random* (gray): the true values of the parameters were not known and not estimated
3. 15_*beta* (dark blue): the parameters were estimated using an ensemble of NNs with 15 members according to the beta method at each DA cycle
4. 15_*beta_init* (light blue): the parameters were estimated using an ensemble of NNs with 15 members according to the beta method only at DA cycle = 0 and then kept constant over time
5. $BNN_0 + BNN_t$ (dark red): the parameters were estimated at DA cycle = 0 using BNN_0 and at DA cycle >0 using BNN_t
6. BNN_0_*init* (pink): the parameters were estimated at DA cycle = 0 using BNN_0 and then kept constant over time

Setup *4* and *6* were chosen because Figures 4.2 and 4.6 indicate that, if the parameters are assumed to be constant over time, the RMSEs of the parameters are lowest for the initial estimation. It might therefore not be necessary to estiamte the parameters at each DA cycle.

4.5.2 Ensemble Size

The ensemble size that is varied in Figures 4.11 and 4.12 refers to the number of ensemble members of the analysis. As expected, the RMSEs of the atmospheric variables decrease for all setups with an increase in ensemble members due to the samples being able to more accurately approximate the true Kalman filtering distribution. For u and h both BNN setups are very close to each other while outperforming both NN setups in all experiments and achieve the same result as the best-case scenario (true) for a large ensemble size of 400. For the rain r $BNN_0 + BNN_t$ needs at least 50 ensemble members to surpass the NN while BNN_0_*init* consistently has the lowest RMSE for all ensemble sizes. The sensitivity of $BNN_0 + BNN_t$ can be explained due to the ensemble size directly controlling the training size for this setup and more training data usually increases the predictive capabilities of a BNN. This is also illustrated in the RMSEs of the parameters which decrease for larger ensemble sizes, especially the rain threshold h_r. For $N_{ens} > 100$ however, the RMSEs of the parameters seem to saturate for $BNN_0 + BNN_t$ which could be caused by the very small network size of only 2 neurons per hidden layer. To test this hypothesis, network size sensitivity experiments were conducted in Figure 4.15. That setup *4* and *6* (15_*beta_init* and BNN_0_*init*) don't show any sensitivity in the parameters is not surprising as these setups only use the initial, interpolated observations as input to estimate the parameters and are therefore independent of the analysis state. It is surprising though that also 15_*beta* shows no sensitivity for the ensemble size at all. Furthermore, even though the parameter RMSEs of 15_*beta_init* is lower than that of 15_*beta*, its spread is larger. This results in a lower RMSE, but higher spread of the rain r for 15_*beta_init* compared to 15_*beta*. For an ensemble size of 400 the spread of r for 15_*beta_init* is slightly higher, around $7.2 \cdot 10^{-5}$, than its RMSE which means that the true value of r is contained in the analysis.

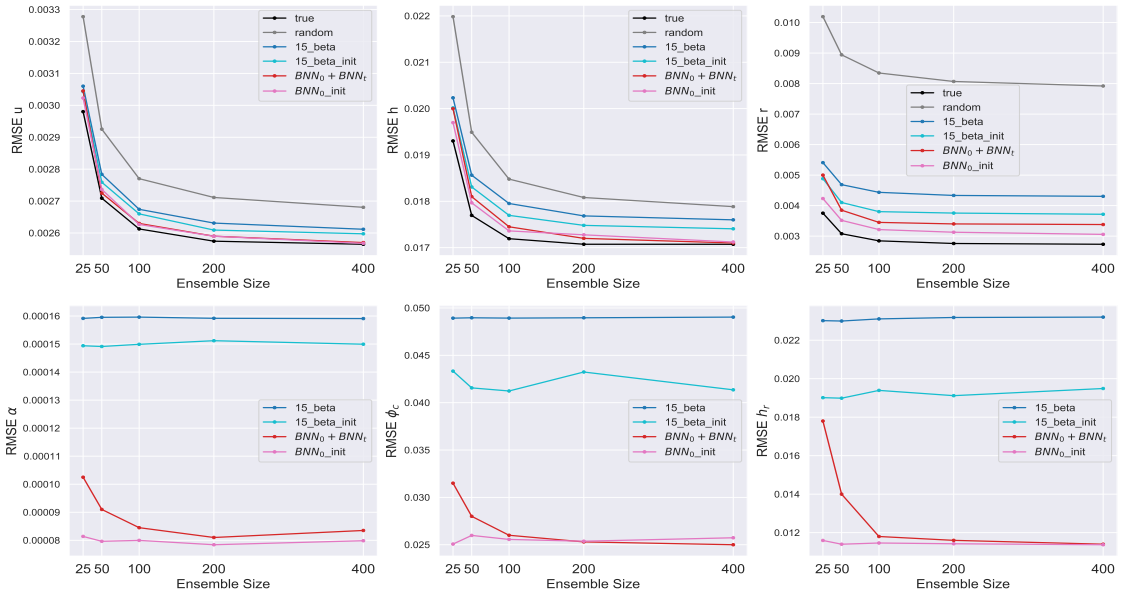

Figure 4.11: RMSEs for atmospheric variable estimates (upper panel) and parameter estimates (lower panel) averaged over last 100 DA cycles of 100 random experiments

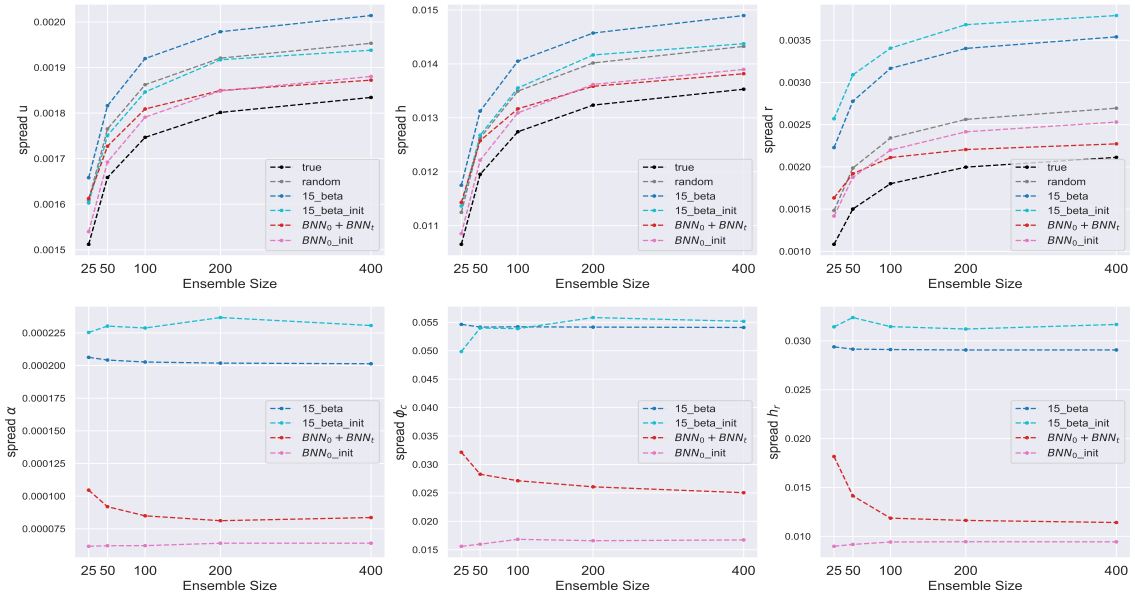

Figure 4.12: Ensemble spread of atmospheric variable (upper panel) and parameter estimates (lower panel) averaged over last 100 DA cycles of 100 random experiments

4.5.3 Observation Coverage

The deep ensemble of NNs and BNN_0 were trained on the full grid, but observations in real-life settings are usually sparse. Therefore, their sensitivity to the *observation size* was investigated, which is here defined as the percentage of observed grid points. Instead of observing only those grid points, whose rain values exceed a cer-

tain threshold as in the other experiments, percentages corresponding to the values of the x-axis in Figures 4.13 and 4.14 of random grid points were observed. The same 6 setups as in section 4.5.2 were used with 100 experiments for each setup and each observation size. For the final scores, the RMSEs of all 100 experiments of the last 100 DA cycles were averaged and plotted against observation size. For the initial parameter estimations, the initial observations were interpolated and used as the input for the ANNs just as in the previous experiments.
The RMSEs of the atmospheric variables, especially u and h, show a strong sensitivity for the observation size and decrease with more observations available. The parameter RMSEs of the deep ensembles of NNs, on the other hand, show almost no sensitivity at all. While the parameter RMSEs of 15_*beta* remain constant for all three parameters, 15_*beta_init* displays a small improvement between $30-40\%$ for ϕ_c and between $60-100\%$ for h_r. The BNNs exhibit a slightly stronger sensitivity compared to the NNs up until around 60% of available observations. This low sensitivity on the observation size indicates that although the ANNs were trained on the whole grid, only some grid points are actually relevant for the ANN's prediction. To investigate this hypothesis the LRP algorithm described in Section 3.3 was utilized in Section 4.6.

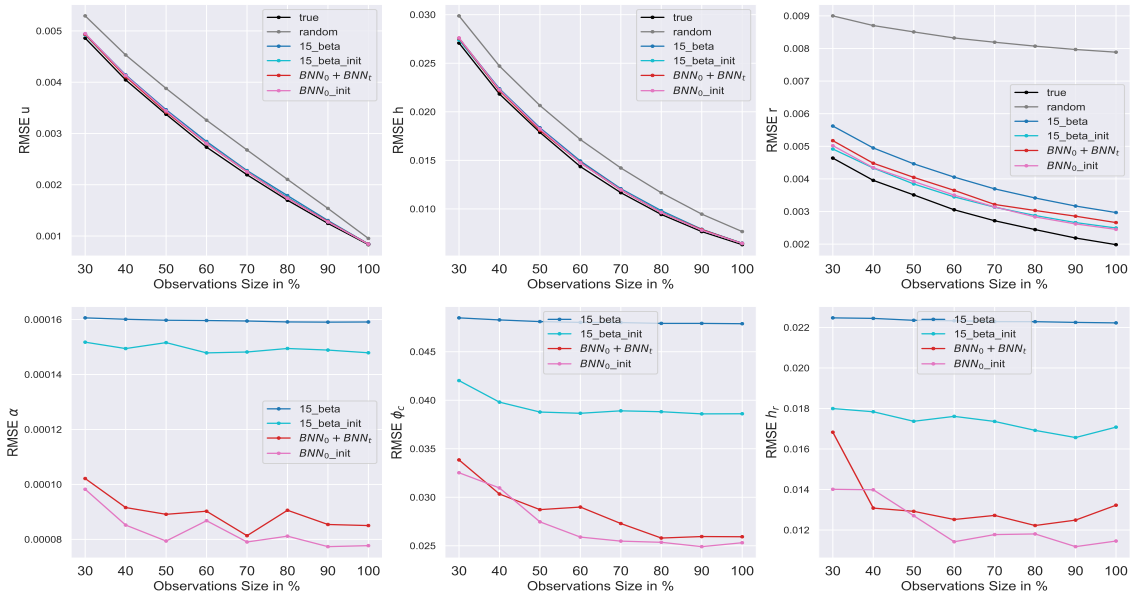

Figure 4.13: RMSE for atmospheric variable estimates (upper panel) and parameter estimates (lower panel) averaged over last 100 DA cycles of 100 random experiments with 50 ensemble members

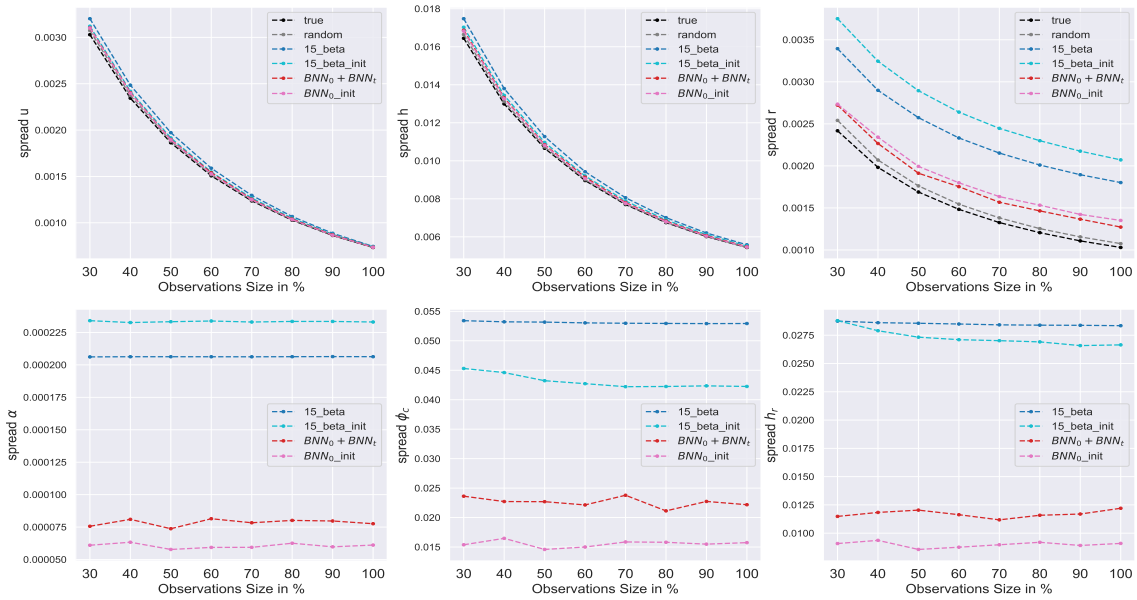

Figure 4.14: Ensemble spread of atmospheric variable (upper panel) and parameter estimates (lower panel) averaged over last 100 DA cycles of 100 random experiments with 50 ensemble members.

4.5.4 Number of Neurons

The experiments discussed in this section investigate the hypothesis, gained from Figure 4.11, that for $N_{ens} > 100$ the predictive ability of BNN_t could be positively influenced by increasing the network size. Here the *network size* is defined as the number of neurons per hidden layer N_h. For all experiments up until this point N_h of BNN_t was set to 2. For the results shown in Figure 4.15 the same 100 experiments as before were run with the state ensemble size set to 200 and varied neurons per hidden layer. Contrary to our hypothesis, the parameter RMSEs increase with an increase in N_h as does the spread. This surprisingly has a positive effect on r, where the RMSE decreases while its spread increases. The velocity u and fluid height h on the other hand show no sensitivity to the network size. Figure 4.15 indicates that not only the accuracy of the parameter estimation but also its spread is relevant for the state error. This in turn highlights the importance of parameter estimation methods that are able to produce relevant distributions such as BNNs.

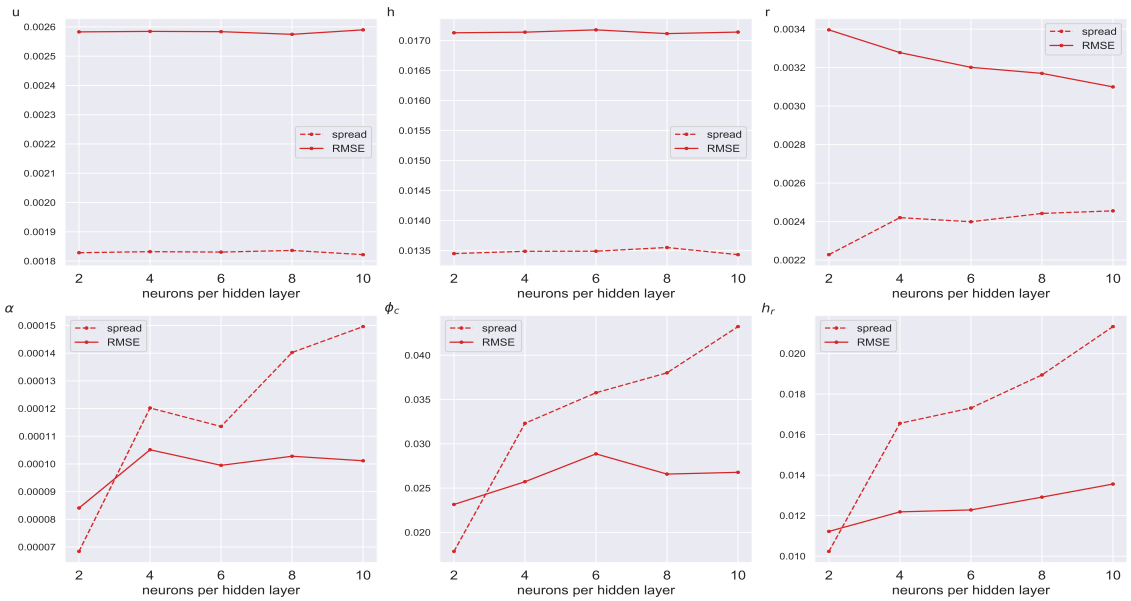

Figure 4.15: RMSE for atmospheric variable estimates (upper panel) and parameter estimates (lower panel) averaged over last 100 DA cycles of 100 random experiments with 200 ensemble members using BNN_t

4.6 Layer-Wise Relevance Propagation

Since, to the best of our knowledge, the LRP algorithm has not been applied to BNNs so far Figures 4.16, 4.17 and 4.20 were produced using only the best performing point estimate NN, which is the same as the one used for Figure 4.1. For Figure 4.16 LRP was applied to all three parameters α, ϕ_c and h_r for 100 inputs and averaged. The inputs used in these experiments are observations taken from the true atmospheric state of fully observed grids. To better compare the NN inputs (observations of the state) with their corresponding LRP heatmaps, the values of u, h, r and the LRP heatmap values were rescaled between 0 and 1 and plotted together as heatmaps. The x-axis represents the 250 grid points while the y-axis has no meaning and just provides a spatial dimension so that the colors of the heatmap are visualized better. Darker red tones correspond to higher values and thus represent grid points that were more relevant for the NN's prediction.

The total relevances plotted in Figure 4.16 are simply the LRP heatmap values of one parameter for a certain atmospheric variable summed and divided by the total sum of LRP heatmap values for that parameter. Even though all experiments conducted so far indicate that r is the most sensitive variable to the parameter estimations, for the NN h is the most relevant variable while u and r have about the same relevance. It is also surprising that even though the results plotted in Figure 4.16 are averaged over many experiments, one can still determine distinct lines over certain grid points. These distinct lines indicate that the NN uses only a small number of grid points to make its prediction instead of the whole grid. Using only a small number of important grid points as the input would in turn decrease the number of learnable parameters and might result in the need for much smaller

training sizes. This finding would also explain why the sensitivity on the observation size discussed in section 4.5.3 is so low. The heatmaps of u and r look very similar to the input indicating that those grid points with strong winds and rain are especially relevant for the NN. The heatmaps of h however look very different from their input and need further analysis to investigate what led the NN to use these grid points to make its prediction. In Figures 4.17 and 4.20 the heatmaps of a single experiment were plotted for all three parameters to investigate which gridpoints were chosen by the NN and if the heatmaps of the three parameters indeed look as similar as Figure 4.16 indicates.

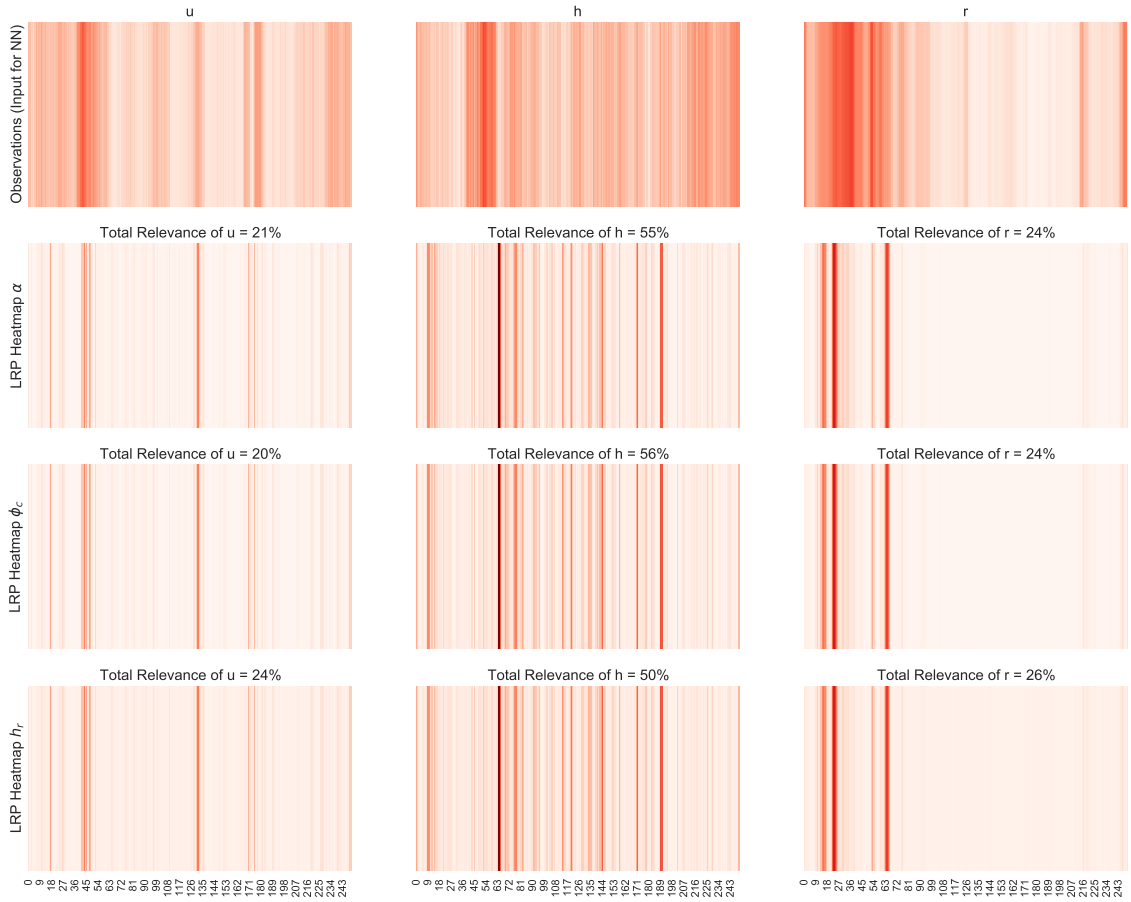

Figure 4.16: Input (first row) and corresponding LRP heatmaps for α (second row), ϕ_c (third row) and h_r (last row) averaged over 100 experiments

Instead of plotting the LRP heatmaps and the atmospheric variable values as heatmaps, they were visualized as graphs with the x-axis corresponding to the grid points and the y-axis corresponding to the rescaled values. The shaded areas represent the values of u, h and r while the red stars are the heatmap values for α, ϕ_c and h_r of the corresponding inputs. Visualizing the LRP outputs this way makes it even more obvious how similar the LRP heatmaps of the three parameters are to each other. Furthermore one can see that grid points with strong winds and rain are very relevant for the NN, although there does not seem to be a distinct relationship between h and its corresponding LRP heatmap. On the other hand, if one plots the LRP heatmap α of h together with the rain as in Figure 4.20, it seems like the

relevant grid points of h are the ones where it is raining. This would explain why simply interpolating the observations and using these to estimate the parameters, as was done for BNN_0_init and 15_beta_init in Figures 4.11 to 4.14, works so well.

Figure 4.17: Rescaled values of fluid velocity u (upper panel), height h (middle panel), rain r (lower panel) and corresponding LRP heatmaps α (red stars) of u,h,r against all 250 grid points for a single experiment

Figure 4.18: Rescaled values of fluid velocity u (upper panel), height h (middle panel), rain r (lower panel) and corresponding LRP heatmaps ϕ_c (red stars) of u,h,r against all 250 grid points for a single experiment

Figure 4.19: Rescaled values of fluid velocity u (upper panel), height h (middle panel), rain r (lower panel) and corresponding LRP heatmaps h_r (red stars) of u,h,r against all 250 grid points for a single experiment

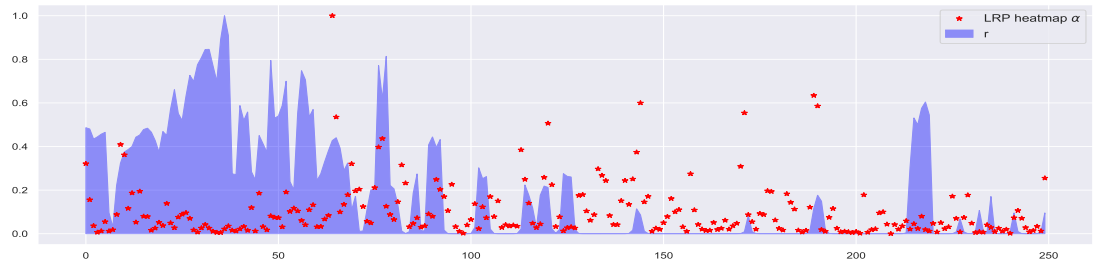

Figure 4.20: Rescaled values of rain r (blue) and LRP heatmap α of h (red stars) against all 250 grid points for the same experiment as Figure 4.17

From a physical standpoint, it is rather surprising that the heatmaps in Figures 4.16 to 4.19 look so similar for all three parameters. To check if this is simply due to the fact that they are predicted simultaneously by a single NN, the same experiments as in Figure 4.16 were repeated with three individually trained NNs, one for each parameter. For the results in Figure 4.21 three training and test datasets were generated: each time keeping two of the parameters constant while varying the parameter that is wished to be estimated. In this setup, there is a clear distinction between the heatmaps of the three individual parameters and also between the heatmaps of Figure 4.16 and Figure 4.21. The relevances shifted from the fluid velocity u and height h towards the rain r for all three parameters, especially for ϕ_c and h_r where the rain is now the most relevant variable. For the rain removal rate α the most relevant variable is still h. While in Figure 4.16 the relevances were concentrated on a few single grid points, they are now more spread out over the grid. To investigate if this spread is due to the averaging over many experiments, the same plot as in Figure 4.17 was created for the three individually trained NNs for one single experiment. When comparing Figures 4.17 to 4.19 with Figures 4.22 to 4.24 it becomes apparent that when the NNs are trained for each parameter individually, almost all rainy grid points are now relevant for the NN's decision instead of just a select few. While α (Figure 4.22) makes use of all thee atmospheric variables, the relevances of ϕ_c and h_r are concentrated on the rain r. Interestingly, although the NNs of ϕ_c and h_r were trained independent of each other, their LRP heatmaps (Figures 4.23 and 4.24) still look quite similar.

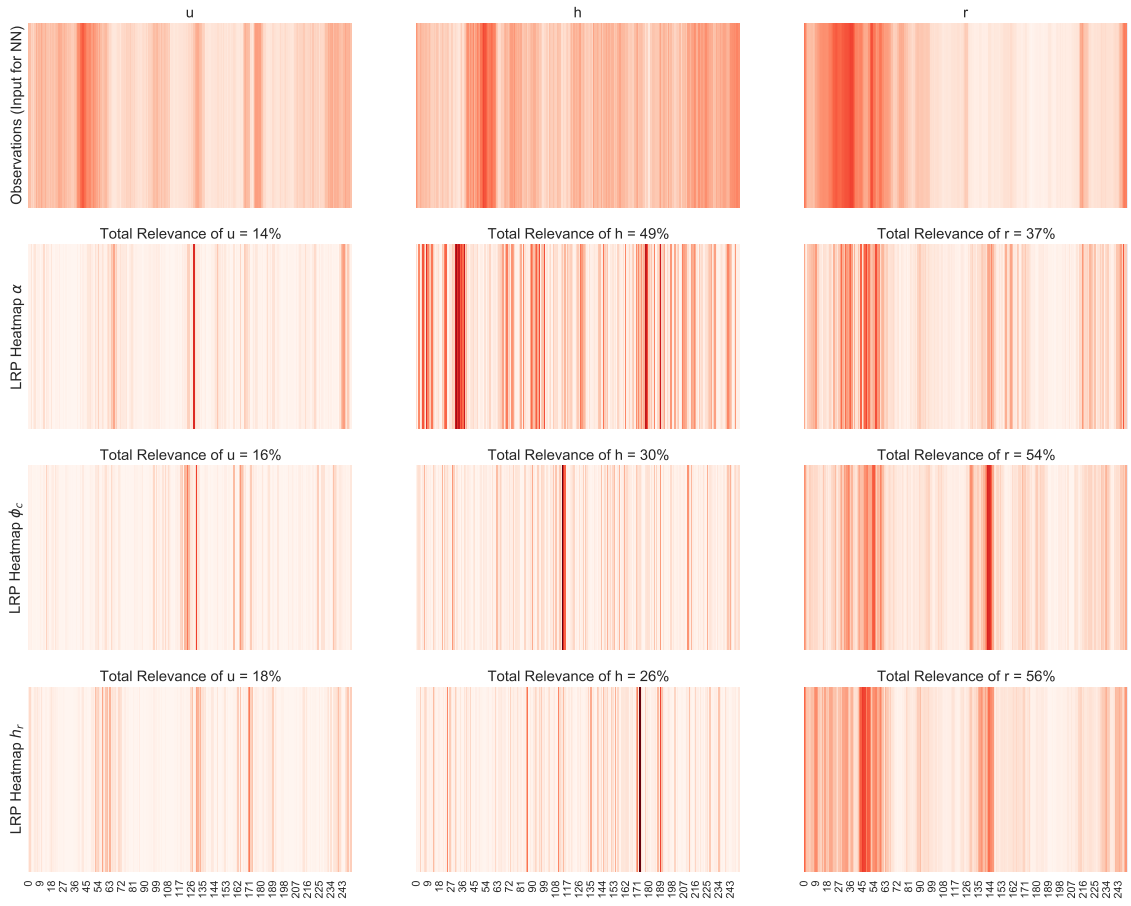

Figure 4.21: Input (first row) and corresponding LRP heatmaps for α (second row), ϕ_c (third row) and h_r (last row) averaged over 100 experiments for 3 individually trained NNs

Figure 4.22: Rescaled values of fluid velocity u (upper panel), height h (middle panel), rain r (lower panel) and LRP heatmap α (red stars) of u,h,r against all 250 grid points for a single experiment of 3 individually trained NNs

Figure 4.23: Rescaled values of fluid velocity u (upper panel), height h (middle panel) , rain r (lower panel) and LRP heatmap ϕ_c (red stars) of u,h,r against all 250 grid points for a single experiment of 3 individually trained NNs

Figure 4.24: Rescaled values of fluid velocity u (upper panel), height h (middle panel) , rain r (lower panel) and LRP heatmap h_r (red stars) of u,h,r against all 250 grid points for a single experiment of 3 individually trained NNs

Chapter 5

Conclusion

In this thesis two types of ANNs were trained to estimate the tunable model parameters of the convective-scale modified shallow water model (Würsch and Craig, 2014). Different methods of integrating the parameter estimates of the ANNs into the data assimilation cycle were developed and studied. Sensitivity experiments of the number of analysis/forecast ensemble members, observation coverage, and BNN size were performed. In the perfect model experiments the NN as well as the BNN were able to decrease the initial state errors of the atmospheric variables by estimating the unknown model parameters. The largest reduction of the initial state error was ultimately found in the rain r. Furthermore, the ANNs investigated here provided tools to quantify the uncertainty of the parameter estimation which increased the spread of the state analysis and forecast while decreasing their RMSEs. Interestingly, even though the rain exhibitted the largest sensitivity on the parameter estimation, the LRP algorithm showed that the fluid height h was the most relevant variable for the NN. In summary, the BNN produced more accurate estimates with more relevant distributions while needing less training time and hyperparameter tuning.

All experiments conducted in this study assumed parameters that are constant in time and space. Future work testing the BNN's ability to estimate local and temporal parameters is therefore required. Furthermore, the training data utilized in this study were snapshots of the grid at one point in time. Alternatively, one could investigate the influence that using a time series of one or more grid points as the BNN's input would have. It should also be noted that the training data, as well as the observations, were generated by the same simplistic model and it is not clear how well the BNN's predictive ability translates to more complex models and real observations. Before testing the BNN in more realistic scenarios, it is necessary to scale down its demand for large training sizes, possibly by reducing the number of input features or number of hidden layers as demonstrated here. Indeed, the LRP heatmaps showed that - at least for the point estimate neural networks - when all parameters are estimated simultaneously, the NN makes use of only a few select grid points. Of course, it would also be necessary to apply the LRP algorithm onto the Bayesian neural network to check if this also holds for the BNN. Another possibility to address these challenges would be to investigate an alternative kind of stochastic ANN. Leinonen et al. (2020) succesfully trained a stochastic generative adversarial network (GAN) to downscale time-evolving images of atmospheric fields from low to high resolution. The GAN trained in Leinonen et al. (2020) consists of convolutional

and recurrent layers and is able to predict images larger than those it was trained on and can predict longer time series than the sequences used for training. This reduces the need for large training sizes and offers the possibility for offline training. Finally, combining ML with DA, as presented in this thesis, can lead to several different hybrid algorithms that mitigate problems for each of the approaches. If it is shown that it is feasible to represent model uncertainty (and observation error) through training an ANN, as for example BNNs, time varying model error and observation error statistics may be included during the DA. Some work has already been done in estimating model errors with ML. Bonavita and Laloyaux (2020) use ANNs to estimate model error tendencies in the Integrated Forecasting System (IFS) of the European Centre for Medium-Range Weather Forecasts (ECMWF) and show that they are able to emulate the main outcomes acquired by the weak-constraint 4D-Var. By utilizing not only the atmospheric variables as the input features but also 'climatological predicators' (Bonavita and Laloyaux, 2020), such as latitude, longitude, time of the day and month, the predictive abilities of the ANNs are greatly enhanced. Providing the ANN with information on the geographical location, diurnal cycle, and seasonal cycle during the training is an interesting approach that could have potential benefits for the parameter estimation problem as well. Furthermore, computational cost can be improved when including ML in the DA. For example, Ruckstuhl et al. (2021) used a convolutional neural network (CNN) to show that a hybrid of a CNN and the EnKF is able to decrease the analysis/background error, equivalent to results obtained by the quadratic programming ensemble (QPEns) (Janjic et al., 2014) but with a reduced computational cost compared to that of the QPEns. Finally, ML approaches can also be improved while using DA by replacing the backpropagation during the training with an adaptive Ensemble Kalman Filter (Trautner et al., 2020).

In the studied test case, with perfect model assumptions and enough training data (100.000 samples), the ANNs were able to estimate the unknown model parameters and quantify their uncertainty more accurately than a simple linear regression, even under sparse and noisy conditions. Including the parameter estimates obtained from the ANNs in the DA cycle resulted in reduced state errors and increased ensemble spreads compared to the case without parameter estimation and unknown parameters.

List of Abbreviations

AI artificial intelligence.

ANN artificial neural network.

BNN Bayesian neural network.

CNN convolutional neural network.

DA data assimilation.

ECMWF European Centre for Medium-Range Weather Forecasts.

EnKF Ensemble Kalman Filter.

GAN generative adversarial network.

GP Gaussian Process.

IFS Integrated Forecasting System.

LeakyReLU Leaky Rectified Linear Unit.

LR linear regression.

LRP Layer-wise Relevance Propagation.

ML machine learning.

NN point estimate neural network.

QPEns quadratic programming ensemble.

ReLU Rectified Linear Unit.

Bibliography

Abbe, C. (1901). "The Physical Basis of Long-Range Weather Forecasts". In: *Monthly Weather Review* 29.

Aksoy, A., F. Zhang, and J.W. Nielsen-Gammon (2006). "Ensemble-based simultaneous state and parameter estimation with MM5". In: *Geophysical Research Letters* 33.12.

Bach, S. et al. (2015). "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation". In: *PLoS ONE* 10.7.

Böhle, M. et al. (2019). "Layer-Wise Relevance Propagation for Explaining Deep Neural Network Decisions in MRI-Based Alzheimer's Disease Classification". In: *Frontiers in Aging Neuroscience* 11.

Bjerknes, V. (1904). "Das Problem der Wettervorhersage, betrachtet von Standpunkt der Mechanik und Physik". In: *Meteorologischen Zeitschrift* 18.

Bonavita, M. and P. Laloyaux (2020). "Machine learning for model error inference and correction". In: *Journal of Advances in Modeling Earth Systems* 12.

Brajard, J. et al. (2020). "Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: A case study with the Lorenz 96 model". In: *Journal of Computational Science* 44, p. 101171.

Cintra, R.S. and H.F. de Campos Velho (2014). "Data Assimilation by Artificial Neural Networks for an Atmospheric General Circulation Model: Conventional Observation". In: *CoRR* abs/1407.4360.

Evensen, G. (2003). "The Ensemble Kalman Filter: theoretical formulation and practical implementation". In: *Ocean Dynamics* 53, pp. 343–367.

Frisinger, H.H (1978). *The History of Meteorology to 1800*. Science History Publications.

Gaspari, G. and S.E. Cohn (1999). "Construction of correlation functions in two and three dimensions". In: *Quarterly Journal of the Royal Meteorological Society* 125.554, pp. 723–757.

Hoffman, M.D. et al. (2013). "Stochastic Variational Inference". In: *J. Mach. Learn. Res.* 14, 1303–1347.

Ioffe, S. and C. Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *CoRR* abs/1502.03167. arXiv: 1502.03167.

Janjic, T. et al. (2014). "Conservation of Mass and Preservation of Positivity with Ensemble-Type Kalman Filter Algorithms". In: *Monthly Weather Review* 142, pp. 755–773.

Jazwinski, A. H. (1970). *Stochastic processes and filtering theory*. Mathematics in science and engineering. Academic Press.

Jospin, L.V et al. (2020). *Hands-on Bayesian Neural Networks – a Tutorial for Deep Learning Users*. arXiv: 2007.06823.

Kingma, D.P. and J. Ba (2014). "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980.

Krasnopolsky, V., M. Fox-Rabinovitz, and A. Belochitski (2013). "Using Ensemble of Neural Networks to Learn Stochastic Convection Parameterizations for Climate and Numerical Weather Prediction Models from Data Simulated by a Cloud Resolving Model". In: *Adv. Artif. Neural Syst.* 2013, 485913:1–485913:13.

Kullback, S. and R. A. Leibler (1951). "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22, pp. 79 –86.

Labach, A., H. Salehinejad, and S. Valaee (2019). "Survey of Dropout Methods for Deep Neural Networks". In: *CoRR* abs/1904.13310.

Labe, Z.M. and E.A. Barnes (2021). "Detecting climate signals using explainable AI with single-forcing large ensembles". In: *Earth and Space Science Open Archive*, p. 40.

Lakshminarayanan, B., A. Pritzel, and C. Blundell (2017). *Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles*. arXiv: 1612.01474.

LeCun, Y., Y. Bengio, and G. Hinton (2015). "Deep Learning". In: *Nature* 521, pp. 436–444.

Leinonen, J., D. Nerini, and A. Berne (2020). "Stochastic Super-Resolution for Downscaling Time-Evolving Atmospheric Fields With a Generative Adversarial Network". In: *IEEE Transactions on Geoscience and Remote Sensing*, 1–13.

Lorenz, E.N. (2005). "Designing Chaotic Models". In: *Journal of the Atmospheric Sciences* 62, 1574–1587.

MacKay, D.J.C. (1992). "A Practical Bayesian Framework for Backpropagation Networks". In: *Neural Comput.* 4, 448–472.

Mcculloch, W. and W. Pitts (1943). "A Logical Calculus of Ideas Immanent in Nervous Activity". In: *Bulletin of Mathematical Biophysics* 5, pp. 127–147.

Mellersh, H.E.L. (1968). *FitzRoy of the Beagle*. Hart-Davis.

Nguyen, T. (2010). "Total Number of Synapses in the Adult Human Neocortex". In: *Undergraduate Journal of Mathematical Modeling: One + Two* 3.

O'Gorman, P.A. and J.G. Dwyer (2018). "Using Machine Learning to Parameterize Moist Convection: Potential for Modeling of Climate, Climate Change, and Extreme Events". In: *Journal of Advances in Modeling Earth Systems* 10.10, 2548–2563.

Rasmussen, C.E. and C.K.I. Williams (2006). *Gaussian Processes for Machine Learning*. MIT Press, p. 248.

Rasp, S., M.S. Pritchard, and P. Gentine (2018). *Deep learning to represent sub-grid processes in climate models*.

Richardson, L.F. (1922). *Weather prediction by numerical process*. Cambridge University Press.

Ruckstuhl, Y., T. Janjić, and S. Rasp (2021). "Training a convolutional neural network to conserve mass in data assimilation". In: *Nonlinear Processes in Geophysics* 28.1, pp. 111–119.

Ruckstuhl, Y. M. and T. Janjić (2018). "Parameter and state estimation with ensemble Kalman filter based algorithms for convective-scale applications". In: *Quarterly Journal of the Royal Meteorological Society* 144.712, pp. 826–841.

Ruiz, J.J., M. Pulido, and T. Miyoshi (2013). "Estimating Model Parameters with Ensemble-Based Data Assimilation: A Review". In: *Journal of the Meteorological Society of Japan. Ser. II* 91.2, pp. 79–99.

Rusak, E. et al. (2020). *A simple way to make neural networks robust against diverse image corruptions*. arXiv: 2001.06057.

Toms, B.A., E.A. Barnes, and I. Ebert-Uphoff (2020). "Physically Interpretable Neural Networks for the Geosciences: Applications to Earth System Variability". In: *Journal of Advances in Modeling Earth Systems* 12.9.

Trask, A., D. Gilmore, and M. Russell (2015). "Modeling Order in Neural Word Embeddings at Scale". In: *CoRR* abs/1506.02338.

Trautner, M., G. Margolis, and S. Ravela (2020). "Informative Neural Ensemble Kalman Learning". In: *CoRR* abs/2008.09915.

Varma, V. (2020). *Embedded methods for feature selection in neural networks*. arXiv: 2010.05834.

Würsch, M. and G.C. Craig (2014). "A simple dynamical model of cumulus convection for data assimilation research". In: *Meteorologische Zeitschrift* 23.5, pp. 483–490.

Yadav, N., S. Ravela, and A.R. Ganguly (2020). *Machine Learning for Robust Identification of Complex Nonlinear Dynamical Systems: Applications to Earth Systems Modeling*. arXiv: 2008.05590.

Yuval, J. and P.A. O'Gorman (2020). "Stable machine-learning parameterization of subgrid processes for climate modeling at a range of resolutions". In: *Nature Communications* 11.1.

Declaration

I herewith declare that I have composed the present thesis myself and without use of any other than the cited sources and aids.

Munich, 18.06.21

Erklärung

Hiermit erkläre ich, die vorliegende Arbeit selbständig verfasst zu haben und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel benutzt zu haben.

München, 18.06.21