Department of Computer Science

Training methods for climate and neural network models

Mudassar Abbas

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 209/2018

Training methods for climate and neural network models

Mudassar Abbas

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Science, at a public examination held at the lecture hall AS2 at the Aalto University School of Science (Espoo, Finland) on 15 November 2018 at 12.

Aalto University
School of Science
Department of Computer Science
Complex Systems

Supervising professor
Prof. Kimmo Kaski, Aalto University, Finland

Thesis advisor
Dr. Jyri Kivinen, Aalto University, Finland

Preliminary examiners
Prof. Teemu Roos, University of Helsinki, Finland
Prof. Tapio Pahikkala, University of Turku, Finland

Opponent
Prof. Jukka Heikkonen, University of Turku, Finland

Aalto University publication series
DOCTORAL DISSERTATIONS 209/2018

ISBN 978-952-60-8258-5 (printed)
ISBN 978-952-60-8259-2 (pdf)
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-60-8259-2

Unigrafia Oy
Helsinki 2018

Finland

441 697
Printed matter

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Mudassar Abbas

Name of the doctoral dissertation
Training methods for climate and neural network models

Publisher School of Science

Unit Department of Computer Science

Series Aalto University publication series DOCTORAL DISSERTATIONS 209/2018

Field of research Computer Science

Manuscript submitted 19 June 2018 **Date of the defence** 15 November 2018

Permission to publish granted (date) 18 September 2018 **Language** English

☐ **Monograph** ☒ **Article dissertation** ☐ **Essay dissertation**

Abstract

When modeling complex phenomena in nature and in technological systems, one is often faced with the task of tuning/calibrating the models. In such cases, there typically exists a need for model parameter (and/or meta-parameter) value tuning for more effective modeling performance. Often such cannot be done manually, and in the machine learning approach, the tuning is done in an algorithmic and data-driven manner, and is called model training. The thesis presents studies in which such methods are adopted, in the contexts of climate and artificial neural networks, and proposes novel techniques.

One of the studies is on the suitability of a well-known machine learning method called Bayesian optimization (BO), for parametric tuning of chaotic systems such as climate and numerical weather prediction (NWP) models. The obtained results show that BO is a suitable method for such tuning tasks.

A major desiderata for a trained machine learning model is the ability to generalize well to unseen data, and thus the phenomena such as (so-called) under- and overfitting are to be avoided. In this context, adopting (so-called) regularization methods as part of the model training process has become a standard procedure. In this thesis, we introduce a regularization framework that is shown to have close connections with many existing state-of-the-art regularization approaches. An adversarial variant, derived from the proposed regularization framework, is used for solving a classification task, and the obtained results are compared to those of other regularization methods.

Keywords

ISBN (printed) 978-952-60-8258-5 **ISBN (pdf)** 978-952-60-8259-2

ISSN (printed) 1799-4934 **ISSN (pdf)** 1799-4942

Location of publisher Helsinki **Location of printing** Helsinki **Year** 2018

Pages 124 **urn** http://urn.fi/URN:ISBN:978-952-60-8259-2

Preface

The work presented in this thesis has been carried out in the Department of Computer Science (CS) of the Aalto University between June 2013 and late 2017, with supervision mainly by Professor Erkki Oja and Professor Kimmo Kaski, and with the majority of the obtained funding received from the Finnish Centre of Excellence in Computational Inference Research (COIN) and from the Department of Computer Science.

Although Dr. Alexander Ilin and Prof. Tapani Raiko have been my main instructors, I have also been instructed by Dr. Jyri Kivinen towards the end of the duration of my PhD work. I am greatly thankful to all my supervisors and instructors for the support and guidance provided to me for the PhD work as well as the financial support through this time period. I would also like to thank Professor Juha Karhunen who has at many times helped me out in several matters related to my PhD work. 'Thanks!' to all once again!

I would like to acknowledge and thank my colleagues Dr. Antti Solonen, Dr. Janne Hakkarainen and Professor Heikki Järvinen with whom I worked for almost 2 years on the project called novel advanced mathematical and statistical methods for understanding climate. I would like to thank Alexander Bibov for providing help and support during the project work. I would also like to thank Jaakko Luttinen, especially for his support and helpful discussions related to the project.

I would like to acknowledge and thank my colleagues Antti Rasmus, Mathias Berglund, Pyry Takala, Antti Tarvainen, Vikas Verma, Pablo Alonso Baldonedo, Vikram Kamath, Jelena Luketina, Gerben van den Broeke and Miquel Perelló Nieto (in no particular order) in the Deep Learning and Bayesian Modelling research group for the time well spent during work.

I would like to thank my friends Hadi Soleimany, Xi Chen and Alexander Grigorevskiy at the CS department, and also Khurram Gulzar, Kashif Gulzar, Farhan Ali and Aqeel Hussain at the Aalto University for all the meetings and casual discussions on several matters close-to and far-off work.

It would be impossible to thank all my friends by name who have made life outside work really pleasant and enjoyable during my PhD in Finland but let me at least mention some of them here: Shamshir Abbas, Syed Ali Aamir, Syed

Hussnain Tanveer, Muhammad Qasim, Syed Shabir Kazmi, Muhammad Ali, Muhammad Salman and Syed Hammad Hassan (again in no particular order).

I would like thank my friends and family. I am grateful to my parents for their love and the support they gave me in every study endeavor of my life. I am thankful to my brother and sisters for the love they share with me. Finally, to the love of my life Nosheen Kausar (my wife), thank you for the care, support and patience during these past years.

Espoo, September 2018,

Mudassar Abbas

Contents

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

I Solonen, A., Hakkarainen, J., Ilin, A., Abbas, M. and Bibov, A.. Estimating model error covariance matrix parameters in extended Kalman filtering. *Nonlinear Processes in Geophysics*, 21, 5, 919–927, 2014.

II Abbas, M., Ilin, A., Solonen, M., Hakkarainen, J., Oja, E. and Järvinen, H.. Bayesian optimization for tuning chaotic systems. *Nonlin. Processes Geophys. Discuss.*, 1, 1283–1312, 2014.

III Abbas, M., Ilin, A., Solonen, M., Hakkarainen, J., Oja, E. and Järvinen, H.. Empirical evaluation of Bayesian optimization in parametric tuning of chaotic systems. *Int. J. Uncertainty Quantification*, 6, 6, 467–485, 2016.

IV Abbas, M., Kivinen, J. and Raiko, T.. Understanding regularization by virtual adversarial training, ladder networks and others. In *International Conference on Learning Representations (ICLR) Workshop track*, Puerto Rico, May 2016.

Author's Contribution

Publication I: "Estimating model error covariance matrix parameters in extended Kalman filtering"

The present author participated in the experimental work and computer simulations, discussed the results and conclusions, and participated in writing the paper with the co-authors.

Publication II: "Bayesian optimization for tuning chaotic systems"

The present author defined the problem together with the co-authors, did the implementation and the experimental work and wrote the paper together with the co-authors.

Publication III: "Empirical evaluation of Bayesian optimization in parametric tuning of chaotic systems"

The present author did the coding and the experimental work and wrote the paper together with the co-authors.

Publication IV: "Understanding regularization by virtual adversarial training, ladder networks and others"

The present author participated in proposing the research work, participated in the mathematical analysis of the problem and wrote the paper together with the co-authors.

Notation

In the thesis, the author has tried to use a consistent notation, although some times they may be different from what is the convention. The notation that is repeatedly used throughout the thesis is presented here to help the reader while most of the text has self-contained description of the used notation as well.

Vectors, which are always assumed to be column vectors, are denoted by bold lower case letters such $\mathbf{x}$, $\mathbf{y}$, $\mathbf{f}$, etc. Scalers are denoted by non-bold lower case letters such as x, y, f, etc. Matrices are denoted by bold upper case Roman or Greek letters such as $\mathbf{Q}$, $\mathbf{\Sigma}$, etc. Lower case Greek letters can denote scalar variables and parameters such as μ and σ denote scalar variables and λ denotes parameters. An exception is $\boldsymbol{\theta}$ which denotes a set of parameters of a model, in vector form. A superscript T denotes the transpose of a matrix or vector. Individual components of a vector are indicated by a subscript, for example, x_i denotes the i-th component of the vector $\mathbf{x}$. A subscript of the form $1:k$ indicates a sequence of values, for instance, $\mathbf{y}_{1:k}$ denotes a sequence of values of the vector $\mathbf{y}$, from the 1-st to the k-th step in the sequence.

Functions, regardless of the dimension of their output, are denoted by non-bold letters such as $f()$ and $\mathcal{K}()$.

1. Introduction

In this chapter, the motivation for the thesis work is discussed together with giving a brief overview of the thesis. The contributions of the thesis are discussed and the structure of the thesis is given in the last section of this chapter.

1.1 Motivation and overview

In order to model phenomena of inherently complex natural and technological systems, one is often using a machine learning based approach, for example, an artificial neural network. For effective modeling performance, the model may need to have a very large number of degrees of freedom, which are dictated by the parameterization (/tunable parameters) the model assumes. A key goal of model fitting is to find values for the tunable parameters of the model so as to maximize the model's generalization performance; the generalization performance relates to the model's degree of modeling effectiveness in modeling problem-related and -representative data that has not been used in tuning the parameter values, i.e. in the model fitting/training process.

The model's parameters (all of which relate to the model specification) are, and often need, to be divided into two distinct types, regular (/learned/optimized) parameters, and meta-parameters, for a training process; values of the parameters are tuned with the use of training data during the training process, where as those of the meta-parameters are not. The training algorithm that realizes the training process may be a parametrized one involving parameters that are independent from the model specification and to include meta-parameters. For methods where the model is iteratively trained, there may be adaptiveness in the values of the meta-parameters during the training process, for example updates happening at each iteration. As an example, an adaptive training-algorithm's meta-parameter could be used to control the dynamics of the training process, and a fixed model's meta-parameter could be used to define the model's capacity. The training process may also be nested, with (say model's) parameters being of meta-type in an inside-loop training algorithm and being of regular-type in a training algorithm that is in an outer-loop. Classical examples of meta-

parameters are the number of clusters in the k-NN clustering (see e.g., Duda et al., 2001, pg. 174), the number of layers and number of units per layer in the artificial neural network (see e.g., Bishop, 1995), or the learning rate when using the gradient descent method (see e.g., Bishop, 2006, pg. 240).

There are two approaches for specifying values for (meta-) parameters: either manually or automatically. Traditionally, the optimal values for the model's meta-parameters may be defined using expert knowledge by rules-of-thumb (Hinton, 2012; Hsu et al., 2003). However, manual tuning seems feasible only when there are a small number of the meta-parameters and/or joint configurations of their values to manually search, and the training of the model's regular parameters given a (single) joint configuration of the meta-parameter values is computationally cheap. Still, the manual tuning is often tedious and an automated tuning solution could be a preferable one. Recently, developing such automated solutions has been one of the research focus areas in machine learning (Bergstra and Bengio, 2012; Hutter et al., 2011; Sutskever et al., 2013). This research focus is especially on optimizing deep neural networks where a plethora of new training methods have introduced additional meta-parameters with effective values to be resolved. The training of deep neural networks's regular parameters given a typical model parameterization is expensive in terms of computational time. Therefore, methods for resolving the appropriate values for the meta-parameters need to be computationally efficient. As a solution to this problem, Bayesian optimization (BO) — a powerful method for finding the extrema of expensive-to-evaluate objective functions (Brochu et al., 2010) — has gained a keen interest from the machine learning community in recent years due to its successful application to a number of problems (Boyle, 2007; Brochu et al., 2010; Frean and Boyle, 2008; Hutter et al., 2013; Osborne et al., 2009; Snoek et al., 2012).

Similarly in climate modeling, meta-parameter estimation is a well-known problem (Jackson et al., 2004; Urban and Fricker, 2010; Järvinen et al., 2010; Neelin et al., 2010; Schirber et al., 2013; Hauser et al., 2012). Evaluation of the objective function requires computationally expensive model simulations, typically requiring several days to complete. Therefore, motivated by the success of BO in complex and computationally expensive problems in machine learning, in this thesis we have applied and studied the effectiveness of the BO approach for climate and numerical weather prediction models.

Another important aspect in learning models from data is that we need to avoid model settings in which the model is not able to generalize well to unseen data settings, as in under- and overfitting settings. This could be achieved through training techniques that use the concept of regularization — any method that improves the model generalization falls under the regularization umbrella (see Goodfellow et al., 2016, sec. 5.2) — commonly adopted in training deep learning neural networks (see Goodfellow et al., 2016, chp. 7). In this context, BO and similarly other meta-parameter optimization methods can even be used to tune the amount of regularization via the regularization meta-parameters. With

the advancement in computing power and the amounts of available data, the deep neural networks or deep learning have been able to achieve state-of-the-art results in many complex problem domains such as image classification, image recognition and in speech recognition (LeCun et al., 2015). These advancements have enabled researchers to often use a rule-of-thumb of specifying the models to have so many tunable parameters so that the models would be able to clearly overfit to the training data but use regularization methods to avoid it (Srivastava et al., 2014). Many regularization methods have been introduced in this context, especially, for deep learning such as the Dropout (Srivastava et al., 2014), Maxout Networks (Goodfellow et al., 2013) and Generative Adversarial Networks (Goodfellow et al., 2014b).

The idea of adversarial examples originated from a surprising discovery (Szegedy et al., 2014). In this study the authors discovered that by adding small perturbations to the training examples one could find the blind spots of the neural networks. Any trained neural network misclassified images with small imperceptible perturbations although the same images were part of the training examples. Later, adversarial training was presented as an effective way of regularization by Goodfellow et al. (2014b). Recently, the method of adversarial training has become a hot research topic in the deep learning community, see for example (Kurakin et al., 2016; Goodfellow et al., 2014a; Salimans et al., 2016; Makhzani et al., 2015).

In this thesis, we present a regularization framework and study the connections between related regularization techniques such as adversarial training. We use the proposed regularization framework as a variant of adversarial training to train the Ladder Networks (Rasmus et al., 2015) and present the results for a widely used benchmark dataset known as the MNIST.

1.2 Contributions of the thesis

In this thesis, the contributions of the following four articles are summarized. Publication I presents an approach based on the Filter Likelihood technique that is used to calibrate the model error covariance matrix in a data assimilation process. Two different representations (parameterizations) of the model error covariance matrix are formulated. In the results, the effect of calibrating the model error covariance matrix parameters on the forecast error are studied. The two-layer quasi-geostrophic (QG) model (Pedlosky, 1987) was used in the experiments. The main results of the study are that the importance of the model error covariance matrix calibration depends on the quality of observations, and that the estimation approach yields a well-tuned model in terms of the accuracy of the state estimates and model predictions. The studied approach can be used as a tool to calibrate the model error parameters in also many other climate models.

Publication II presents Bayesian optimization (BO) as a tool for parametric

tuning of chaotic systems. First, the BO is applied for tuning the values of the parameters that are used to parameterize the model error covariance matrix in the Filter Likelihood scheme for the two-layer quasi-geostrophic (QG) model. The parameterization of the model error covariance matrix in the QG model is formulated. Secondly, the BO is used for tuning the values of the parameters used in the polynomial parameterization in the Lorenz 95 (Lorenz, 1995) model. The results show that the BO is able to tune both the studied models with a low number of objective function evaluations and without the need of any gradient information.

Publication III studies the suitability of the Bayesian optimization (BO) approach for tuning climate and numerical weather prediction models. The BO is applied to two benchmark models, namely the two-layer quasi-geostrophic (QG) model and the Lorenz 95 model. For the Lorenz 95 model, the experiments consist of the comparison of the BO method with the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) method (Hansen and Ostermeier, 1996), as well as the formulation of the performance metric to test the accuracy of the tuned models in terms of the forecast error, and the lead time analysis. The BO approach is able to find the optimal meta-parameter values in both of the tuning tasks with a low number of objective function evaluations. The results show that the BO approach is a suitable technique for tuning expensive-to-evaluate objective functions in climate and numerical weather prediction (NWP) systems.

In Publication IV, a regularization framework is presented, where an original clean data point and a nearby point are fed through a mapping, which is then penalized by the Euclidean distance between the corresponding outputs. The theoretical connections between the studied regularization framework and many existing regularization methods are shown. It is demonstrated that the studied approach is a stochastic estimate of penalizing the Frobenius norm of the Jacobian of the mapping parameters, which generalizes the noise regularization, and is a simplification of the canonical regularization term by the Ladder Networks. In addition, the connection to the virtual adversarial training (VAT) method (Miyato et al., 2016) is investigated and it is shown how the VAT can be interpreted as penalizing the largest eigenvalue of a Fisher information matrix.

In Chapter 4, we present the results for the adversarial variant of the regularization framework studied in Publication IV. The framework is used to train the Ladder Networks and the results are presented for the MNIST classification task.

1.3 Structure of the thesis

The thesis provides the basic concepts required to understand the methodology used for the studied research problems, an introduction to the methodology, and highlights the results of the research. In Chapter 2, we discuss the general machine learning concepts relevant to the thesis. Chapter 3 presents an intro-

duction to the Bayesian Optimization method and related topics. We discuss the case-studies used in our research work and present some of the results. Chapter 4 presents the regularization methods and adversarial training where we discuss the connections between regularization schemes and show the results. In Chapter 5, we provide the conclusions and future directions of the research in this thesis work.

2. Machine learning basics

Machine learning may be classified e.g. as a collection of methods that may be used to algorithmically extract regularities or patterns from data (Murphy, 2012; Bishop, 2006). Identifying patterns from huge datasets is a non-trivial task that is practically impossible to solve in a limited time-frame without the use of computational methods and fast enough computers. Once useful patterns are known they help us convert data to information and knowledge i.e., help make informed decisions based on data. Moreover, these patterns can be used to understand the process that generated the data and to make future predictions.

While machine learning has been widely used in data analysis in many branches of science, its most important role has been to build artificial intelligence. For a machine to be *intelligent* it should at least be able to adapt to the changing environment that it is a part of (Alpaydin, 2014). If it is able to adapt then the system designer may not have to provide any further assistance to the machine. Many tasks are too difficult to solve using conventional computer programs and often human experts are involved in the decision making process.

Some common machine learning tasks include: classification, regression, anomaly or outlier detection, denoising and imputation of missing values; see (Bishop, 2006; Alpaydin, 2014; Murphy, 2012) for details on the different kinds of machine learning tasks. An everyday example of the kind of problem for which machine learning could be used is prediction of rain at a specific time and location. For this a machine learning approach would require to tune the parameters of an adaptive model on a set of N vectors $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, called the training data set. Each vector $\mathbf{x}_n$ consists of numerically valued weather measurements such as numerical values of temperature, wind, humidity, and pressure, at some particular time point in the past. Each such training observation vector ($\mathbf{x}_n$) would have an associated label y_n which would indicate whether it rained or not at the location then. The model parameters are tuned so that the model would map an input $\mathbf{x}$ to a target output $\mathbf{y}$ effectively, using the training data. The training phase might generate a functional approximation in the form of $y = f(\mathbf{x})$. After training is complete the performance is usually checked on unseen observations, called the test data set. The performance of the method over the training and test set is given by the training error and the test error, re-

spectively. The ability of the method to correctly classify the observations, where the goal is to reduce the test error at the expense of increased training error, is known as generalization (Goodfellow et al., 2016). The strategies collectively adopted for this are known as regularization methods. Regularization can be defined as any kind of modification made to a machine learning method that improves its generalization (Goodfellow et al., 2016).

2.1 Machine learning methods

In general, machine learning methods can be broadly speaking categorized as supervised and unsupervised learning (Murphy, 2012). More recently, there has been the introduction of semi-supervised learning (Chapelle et al., 2006). Another type of machine learning is called reinforcement learning, however, that is beyond the scope of this thesis work. In the following sections a brief introduction is given to the three types of machine learning: supervised, semi-supervised and unsupervised.

2.1.1 Supervised training methods

In supervised learning, the task is to learn to associate some input with some corresponding output, given a labeled dataset of input-output pairs $\{x, y\}$ (Murphy, 2012). Each input example in the dataset is associated to an output label or target. The training method learns a mapping between the input examples and the associated labels. Both classification and regression are examples of supervised learning tasks (Alpaydin, 2014). The output of a classifier is of different form than the output of a regression model. In classification, normally the outputs are defined to be categories from a finite set of discrete values (e.g. $y \in \{1, \ldots, M\}$, where M denotes the number of possible classes in the classification task). In a regression task, the outputs are real-valued (Murphy, 2012).

2.1.2 Unsupervised training methods

More often, it is difficult to find a human supervisor or automatically generate the labels (Goodfellow et al., 2016). In such cases, unsupervised learning is used to find the structure or regularities, only from the input data (Murphy, 2012; Alpaydin, 2014). Usually, unsupervised learning is used to discover knowledge about the source that generated the data (Goodfellow et al., 2016). This is also known as density estimation in statistics (Alpaydin, 2014). Clustering methods are an example of unsupervised learning where the aim is to group the training set examples based on some suitable distance measure. Statistical methods such as principal component analysis (PCA) and independent component analysis (ICA) are also examples of unsupervised training methods (Alpaydin, 2014).

2.1.3 Semi-supervised training methods

Between supervised and unsupervised learning there is semi-supervised learning, development of which accelerated in the 1970s (Chapelle et al., 2006). A common scenario that arises in machine learning is the availability of few labels while majority of the data being unlabeled. With semi-supervised learning both the labeled and unlabeled data are used to solve a task that is originally a supervised learning task. A natural question to ask then is how useful is the unlabeled data? and the goal is to make as much use of unlabeled information as possible (Goodfellow et al., 2016). This is a very effective strategy because data is naturally in unlabeled form. Although manual labeling is possible it is more often very tedious.

2.2 Machine learning concepts

All machine learning methods can be collectively understood with similar concepts. A few concepts that are central to understanding machine learning training methods are presented in this section. More rigorous discussions about machine learning concepts can be read from e.g. the following books Alpaydin (2014); Bishop (2006); Murphy (2012).

2.2.1 Generalization

A major challenge for machine learning methods is the improvement of the generalization ability. Generalization is the ability of a method to perform well on unseen data that was not part of training (Goodfellow et al., 2016). The two main settings of a machine learning model restricting it from generalization are overfitting and underfitting. Either of them may occur when a machine learning method is used to fit a model to the training data.

Underfitting occurs when the model is too smooth or ignores the finer details in the data. This results in a training error which is not small enough. This kind of problem is often easily solvable, usually it can be solved by increasing the model complexity. Overfitting occurs when the model is unable to avoid unnecessary details in the input data, one reason for which might be the presence of some noise source (Murphy, 2012). As a result, the gap between the training error and the test error is too large (Goodfellow et al., 2016). A common solution to this problem is increasing the amount of training data, which, however, works only to a certain point (Alpaydin, 2014). The goal in generalization is then to make training error small and at the same time make the gap between training and test error to be small (Goodfellow et al., 2016); Fig 2.1 provides an illustration related to the gap.

Usually, overfitting and underfitting can be controlled by meta-parameters that alter the model capacity. Roughly speaking, the model capacity stands for

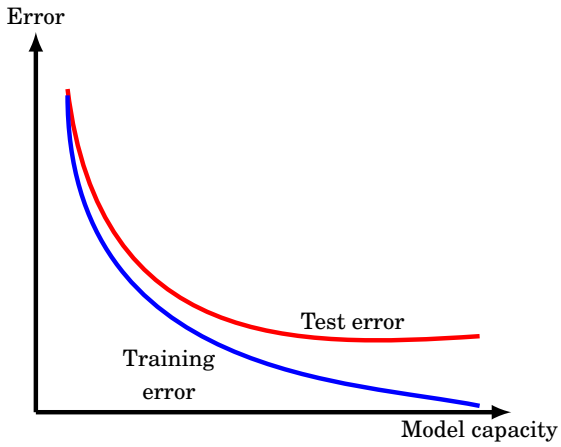

Figure 2.1. The generalization gap of a machine learning model is represented by the gap between the test error and the training error.

the ability of a model to fit a wide variety of functions to the data (Goodfellow et al., 2016). A low capacity model may not be able to properly fit the data while a high capacity model may overfit and then capture unnecessary details in the data.

2.2.2 Validation sets

In order to measure the generalization ability of a method during training a validation set is required. The validation set could not be the same as the test set because this could lead to the point where increasing the model capacity would not improve the performance of the model, hence the model would reach what is known as its maximum capacity (Goodfellow et al., 2016). The validation set is usually obtained when training data is split into two separate sets with a common rule to split being the 80-20 rule. This means that 80% of the data is used as training input and 20% is used for testing the model during training or prior to final model selection. In model selection, the goal is to select the model that performs best on the validation set. Another common approach is to use several validation sets from the same training data, this is known as k-fold cross validation (Bishop, 2006; Murphy, 2012). Although validation sets are used for model selection they are also being used for selection of the model meta-parameters. See (see Goodfellow et al., 2016, sec. 5.3) for a discussion on validation sets and meta-parameters.

2.2.3 Regularization

There are a wide variety of approaches for generalization in machine learning. Any approach that modifies a machine learning method in order to improve its

generalization ability is called a regularization method (Goodfellow et al., 2016). For example, selection of the capacity of a model is itself a regularization method. In this regularization scheme, the size of the hypothesis space of solutions is determined so that the model best fits the data. Another stronger regularization scheme is where the preference is given to a specific function over other functions in the hypothesis space (Goodfellow et al., 2016).

Generally, regularization is implemented by the introduction of a penalty term into the error function (Alpaydin, 2014; Bishop, 2006). The training is then done on this modified error function, for example, neural networks are often regularized by adding the so-called weight decay penalty to the error function (Bishop, 2006). The goal of weight decay is to encourage the weights of the neural network to have small norms, usually, the penalty based on the L^2 norm (the Euclidean norm) of the weights is considered. In the L^2-case, a scaled version (with a positive scalar, say $\lambda/2$) of the norm squared is considered as the weight decay penalty. The weight decay meta-parameter λ then controls the amount of penalization. If λ is zero it means that there is no preference on regularization, a small value allows the model to tend towards overfitting and a large value allows the model to tend towards underfitting.

2.3 Machine learning models

In this section, we present some machine learning models that have been used in this thesis work.

2.3.1 Feed-forward neural networks

Feed-forward neural networks are the most common type of neural networks, and include well-known classes of neural networks such as so-called multi-layer perceptrons (MLPs). Feed-forward neural networks typically take vectors $\mathbf{x}$ as input and map them to outputs y. Therefore, such neural networks can be represented by a function $y = f(\mathbf{x};\boldsymbol{\theta})$ that approximates the relationship between the input and output. Usually, these functions are cascaded such as

$$y = f^3(f^2(f^1(\mathbf{x}))), \tag{2.1}$$

to imply a multi-layer network. In this example, f^1 is the function applied to the outputs of the first layer, f^2 is the function applied to the outputs of the second layer and f^3 is the function applied to the outputs of the third layer. The size of the input layer is often the same as length of the input $\mathbf{x}$. The hidden layer's size is a free meta-parameter and the output layer size depends on the output itself. The layer size stands for the number of units and each unit consists of an activation function $f(\cdot)$ that maps the input to the next layer. The units are also called neurons since the early neural networks were biologically inspired by how the brain learns (see e.g., Bishop, 2006; Goodfellow et al., 2016). Typically, the

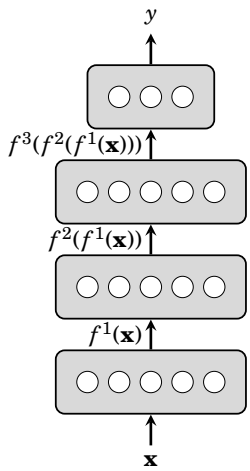

Figure 2.2. An illustration of the feed-forward neural network model. The rounded rectangles represent the layers. The size of the layer is equal to the number of neurons (represented with circles) in the layer.

activation function is the same for the whole network or at least for all the units in a single layer of the network. Commonly used activation functions are the linear rectifier and the logistic function (see e.g., Bishop, 2006; Goodfellow et al., 2016). The Fig. 2.2, shows an illustration of a feed-forward neural network.

The layers of the neural network are connected to form a graph like structure. In a fully-connected neural network, each unit in a layer is connected to every other unit in the previous and next layer to it. The connections are represented by weights and biases, together they are called as parameters of the network (see e.g., Bishop, 2006; Goodfellow et al., 2016). Backpropagation (Rumelhart et al., 1986) or simply backprop is an efficient method for updating the parameters by computing the derivatives of the error function w.r.t the weights and biases. The name backpropagation is given because information flows in the backward direction through the network (Goodfellow et al., 2016). Usually for training the neural network, backpropagation is combined with other methods such as stochastic gradient descent.

Autoencoder

An autoencoder is a type of feed-forward neural network where the purpose is the reconstruction of the input of that neural network. The autoencoder often consists of two parts, namely, the encoder part and the decoder part. The encoder maps the inputs $\mathbf{x}$ to a hidden representation $\mathbf{h} = f(\mathbf{x})$. The decoder maps $\mathbf{h}$ to the output $\hat{\mathbf{x}} = g(\mathbf{h})$. Figure. 2.3 shows an autoencoder with a single hidden layer that generates the representation or the code.

The autoencoder was introduced as a method for dimensionality reduction in the 80's (Baldi and Hornik, 1989). The autoencoder can be trained using the

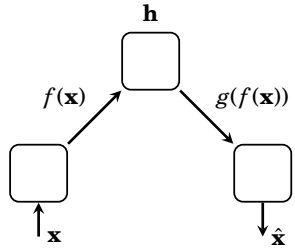

Figure 2.3. An illustration of the autoencoder model with a single hidden layer.

same methods as used by other feed-forward neural networks such as back-propagation. The training criteria for autoencoders are deliberately designed in such a way that the autoencoder is unable to perfectly copy the input to the output $g(f(\mathbf{x})) = \mathbf{x}$ because that would result in poor generalization of the model (Goodfellow et al., 2016).

It is easy to fail to learn a useful hidden representation or code if the size of the hidden layer of the autoencoder is the same or greater than the input layer size. This is also known as the overcomplete case. The undercomplete case for obtaining only an approximation of the input is by using a hidden layer dimension less than the input dimension, such models are called bottle-neck autoencoders. However, it is difficult to learn useful representations or code from the undercomplete case. Therefore, regularization methods are used in combination with the overcomplete case to learn more rich representations. See the book by Goodfellow et al. (2016) for a detailed discussion about representation learning.

2.3.2 State-space models for climate and weather

In climate and numerical weather prediction (NWP), mathematical models are used to describe complex dynamical systems. In general, such models consist of a set of states and parameters that need to be evaluated over time. Therefore, the state and parameter estimation is a central problem in such systems. The state estimation problem is usually solved using filtering techniques that can be given with a pair of state and observation formulas as follows

$$\mathbf{s}_k = \mathscr{M}(\mathbf{s}_{k-1}) + \mathbf{E}_k \tag{2.2}$$

$$\mathbf{y}_k = \mathscr{K}(\mathbf{s}_k) + \mathbf{e}_k, \tag{2.3}$$

where $\mathbf{s}_k$ is the state of the model, $\mathbf{y}_k$ is the observation vector, $\mathscr{M} : \mathbb{R}^m \longrightarrow \mathbb{R}^m$ is the dynamical state-space model which can be implemented by a solver of partial differential equations and $\mathscr{K} : \mathbb{R}^m \longrightarrow \mathbb{R}^n$ is the observation operator. $\mathbf{E}_k$ and $\mathbf{e}_k$ are noise terms which account for model imperfection and observation noise. In climate science applications, model parameters $\boldsymbol{\theta}$ usually appear in the formulation of $\mathscr{M}$ or/and they can govern the distribution of the model error term $\mathbf{E}_k$.

The states $\mathbf{s}_k$ are the representation of the dynamically changing conditions in the physical space and the parameters $\boldsymbol{\theta}$ are static variables that control the system. The challenge is two fold: estimation of the unknown state $\mathbf{s}_k$ at given time instant k and finding optimal parameters $\boldsymbol{\theta}$ for accurate state estimation.

3. Parametric tuning of climate models

Many climate and numerical weather prediction (NWP) models consist of tuning parameters (meta-parameters). For example, tuning of the closure parameters in climate models is required because many processes are imprecisely modeled due to the restrictive grid used for solving the differential equations (Järvinen et al., 2010; Schirber et al., 2013). Another example is the tuning of the parameters that control the stochastic physics components in the ensemble prediction systems (Leutbecher and Palmer, 2008). The closure parameter estimation problem in Hakkarainen et al. (2013) and the problem of tuning the ensemble prediction systems in Solonen and Järvinen (2013) is solved using the Filter Likelihood method.

The difficulty in tuning the meta-parameters is due to several factors: the high computational cost of running climate simulations, the objective function can be noisy i.e., two evaluations of the objective function with the same meta-parameter values usually results in distinct objective function values, and another source of noise is chaos by which we mean that small perturbations of the meta-parameters can result in significantly different simulation trajectories and therefore significantly different objective function values.

The optimization of expensive-to-evaluate black box objective functions usually relies on building surrogate models such as the response surfaces, space mapping, artificial neural networks and Gaussian processes (GP). In this chapter, we present the Bayesian optimization (BO) method which uses the GP. In BO, the meta-parameter values where the objective function is evaluated are carefully chosen so that we learn as much as possible about the objective function. As a result, the optimum can often be found with a small number of function evaluations.

The Filter Likelihood method is based on the computation of the likelihood using the output of a data assimilation system where the goal is to estimate the dynamically changing state of the model, given incomplete and noisy observations. Later in this chapter the description is given of how the Filter Likelihood method and the BO method combine to tune the climate and the NWP systems. Evaluating the likelihood function requires running the data assimilation process based on a computationally expensive model. The BO method

utilizes the GP based model as a surrogate of the tuned chaotic system. The GP parameters are then computed as part of the optimization process. The experiments consist of two case studies: parametric tuning of a simplified atmospheric model with a noiseless likelihood function where the tuned model is a two-layer quasi-geostrophic model with four tuned parameters that define the model error covariance of the corresponding data assimilation system and parametric tuning of a chaotic system with a noisy likelihood function where the parameterized Lorenz 95 model is used as a test model, similarly to previous studies.

The structure of the rest of this Chapter is as follows: In the following section the basic concepts of the Bayesian optimization (BO) method are described. Section 3.1.1 presents the method of Gaussian processes regression and section 3.1.2 shows two acquisition functions used by BO, namely the expected improvement (EI) and the probability of improvement (PI). The Filter Likelihood method is presented in section 3.2. Some results of the case studies in which the BO approach was used for tuning the meta-parameters are shown in section 3.3.

3.1 Bayesian optimization

In Bayesian optimization (BO), the aim is to find the extrema of the black-box functions, $f : \mathbb{R}^n \to \mathbb{R}$ that are computationally expensive to evaluate (see, e.g., reviews by Brochu et al., 2010; Snoek et al., 2012). Here f that is also known as the objective function, typically, does not have a closed form solution. The objective function f is modeled as a random function and its distribution describes our knowledge of the function, given a set of function evaluations $\{\boldsymbol{\theta}_i, f(\boldsymbol{\theta}_i)\}_{i=1,\ldots,t}$. The posterior distribution over f is handled using the standard Gaussian process methodology which allows for evaluating the mean and variance of objective function values $f(\boldsymbol{\theta})$ in any location $\boldsymbol{\theta}$ at any optimization step t. This information is then used to propose a new input location $\boldsymbol{\theta}_{t+1}$ that is expected to have the largest potential to improve over the current best value of the objective function.

The search of the new point (location) where the objective function has to be evaluated is obtained by optimizing a complementary function called *acquisition function*. The two statistics regularly adopted in design of acquisition functions are the predictive mean and the predictive variance of f at any possible location $\boldsymbol{\theta}$. In designing the new points where the function is evaluated, usually one has to choose between two extremes: sampling from locations of high predicted mean value (*exploitation* strategy) and locations of high uncertainty value (*exploration* strategy). BO provides a tool that is able to automatically trade off between exploration and exploitation, which often yields a reduced number of objective function evaluations needed (Lizotte et al., 2012). It can also prove useful for objective functions with multiple local optima, and noise in the objective function can be handled in a straight-forward manner.

3.1.1 Gaussian processes

The methodology of Gaussian processes (GP) works as a central element for BO, as it is an efficient tool for describing distributions of unknown functions (Rasmussen and Williams, 2006). In this methodology, the prior distribution of f is chosen such that the function values $\mathbf{f}_\theta = [f(\boldsymbol{\theta}_1), \ldots, f(\boldsymbol{\theta}_t)]$ are assumed to be normally distributed:

$$\mathbf{f}_\theta \mid \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{f}_\theta | \mathbf{0}, \mathbf{K}_f) \tag{3.1}$$

where the mean is typically taken to be zero and the covariance matrix $\mathbf{K}_f$ is constructed such that its ijth element is computed using the covariance function $k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j | \boldsymbol{\eta})$ that depends on $\boldsymbol{\theta}_i$, $\boldsymbol{\theta}_j$ and hyperparameters $\boldsymbol{\eta}$. The covariance function k is the key element of the GP modeling: it encodes our assumptions about the behavior of the function f, such as its smoothness properties.

A commonly used covariance function, also adopted in our work, is the squared exponential covariance function:

$$k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j; \boldsymbol{\eta}) = \sigma_f^2 \prod_{d=1}^{D} \exp\left(-\frac{(\theta_i^{(d)} - \theta_j^{(d)})^2}{2\, l_d^2} \right), \tag{3.2}$$

where l_d are the parameters defining the smoothness of the function in each dimension and σ_f^2 is the scaling parameter which specifies the magnitudes of the function values. Both belong to the set of hyperparameters $\boldsymbol{\eta}$.

The BO algorithm works such that at every iteration, the properties of the unknown function f are learned by adapting the hyperparameters $\boldsymbol{\eta}$ to fit well the observed data $\{\boldsymbol{\theta}_i, f(\boldsymbol{\theta}_i)\}_{i=1,\ldots,t}$. This is typically done by maximizing the log marginal likelihood of the hyperparameters $\boldsymbol{\eta}$:

$$\log p(\mathbf{f}_n | \boldsymbol{\theta}, \boldsymbol{\eta}) = -\frac{1}{2} \mathbf{f}_n^T (\mathbf{K}_f + \boldsymbol{\Sigma})^{-1} \mathbf{f}_n - \frac{1}{2} \log |\mathbf{K}_f + \boldsymbol{\Sigma}| - \frac{n}{2} \log 2\pi \,. \tag{3.3}$$

where $\mathbf{f}_n$ are the observed values of the objective function. They are assumed to be noisy such that $\mathbf{f}_n = f(\boldsymbol{\theta}) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. $\boldsymbol{\Sigma}$ is the covariance matrix of the noise in the objective function, which is often parameterized as $\sigma_n^2 \mathbf{I}$ and estimated in the optimization procedure as well. Thus $\boldsymbol{\eta}$ consists of the following hyperparameters $\{\sigma_f, l_d, \sigma_n\}$. Note that σ_n is considered a hyperparameter and optimized only when the observed values of the objective function are noisy. Then, GP is used to evaluate the predictive distribution over the function values $f(\boldsymbol{\theta}_{\text{new}})$ at any new location $\boldsymbol{\theta}_{\text{new}}$. Assuming that the observed values of the objective function are noisy and the noise is Gaussian, the predictive distribution is normal:

$$p(f(\boldsymbol{\theta}_{\text{new}}) | \mathbf{f}_n, \boldsymbol{\eta}) \sim \mathcal{N}\left(\mu(\boldsymbol{\theta}_{\text{new}}), \sigma^2(\boldsymbol{\theta}_{\text{new}})\right) \tag{3.4}$$

with the mean and variance given by

$$\mu(\boldsymbol{\theta}_{\text{new}}) = \mathbf{k}_{\text{new}}^T \left(\mathbf{K}_f + \boldsymbol{\Sigma}\right)^{-1} \mathbf{f}_n \tag{3.5}$$

$$\sigma^2(\boldsymbol{\theta}_{\text{new}}) = k(\boldsymbol{\theta}_{\text{new}}, \boldsymbol{\theta}_{\text{new}}) - \mathbf{k}_{\text{new}}^T \left(\mathbf{K}_f + \boldsymbol{\Sigma}\right)^{-1} \mathbf{k}_{\text{new}}, \tag{3.6}$$

where $\mathbf{k}_{\text{new}} = [k(\boldsymbol{\theta}_{\text{new}}, \boldsymbol{\theta}_1), \ldots, k(\boldsymbol{\theta}_{\text{new}}, \boldsymbol{\theta}_t)]^T$. For more details on training GPs, see, for example, the book by Rasmussen and Williams (2006).

3.1.2 Acquisition functions

The acquisition functions are used to search for a new point (location) $\boldsymbol{\theta}_{\text{new}}$ that is expected to have the ability to improve over the best value of the objective function obtained so far, denoted by

$$\mu^+ = \max_t \mu(\boldsymbol{\theta}_t).$$

At each BO iteration, the new sample is chosen to maximize the value of the acquisition function:

$$\boldsymbol{\theta}_{\text{new}} = \arg\max_{\boldsymbol{\theta}} g(\boldsymbol{\theta}),$$

where

$$g(\boldsymbol{\theta}) = g(\mu^+, \mu(\boldsymbol{\theta}), \sigma(\boldsymbol{\theta})).$$

High values of the acquisition function correspond to the regions where the expected value $\mu(\boldsymbol{\theta})$ of the objective function value is high or where the prediction uncertainty $\sigma(\boldsymbol{\theta})$ is high or both. Deciding which areas have the largest potential is known as the exploration vs exploitation trade off (see, e.g., Jones, 2001).

Two of the most popular acquisition functions, also adopted in our work, are the *probability of improvement* (Kushner, 1964) and the *expected improvement* (Mockus, 1989). The probability of improvement (PI) is formulated as

$$g_{\text{PI}}(\boldsymbol{\theta}) = \Phi(\Delta/\sigma(\boldsymbol{\theta})) \tag{3.7}$$

$$\Delta = \mu(\boldsymbol{\theta}) - \mu^+ - \xi \tag{3.8}$$

where $\mu(\boldsymbol{\theta})$ and $\sigma(\boldsymbol{\theta})$ are defined in Eq. (3.5) and Eq. (3.6), respectively. $\Phi(\cdot)$ is the normal cumulative distribution function. When $\xi = 0$, $g_{\text{PI}}(\boldsymbol{\theta})$ is simply the probability of improving the best value μ^+ by taking a sample at location $\boldsymbol{\theta}$. The problem of using $\xi = 0$ is that PI favors locations that have even a slight improvement over the current best μ^+. This means that in this setting PI has a higher tendency to exploit rather than explore and it practically always gets stuck at a local optimum (Lizotte et al., 2012). The parameter $\xi > 0$ allows for tuning PI in order to reduce this problem. However, the choice of ξ is always subjective, although it has a great impact on the performance. The work of Lizotte et al. (2012) provides a detailed study on the effect of ξ on PI and EI acquisitions.

The expected improvement (EI) is formulated as

$$g_{\text{EI}}(\boldsymbol{\theta}) = \langle \mathbf{f}_\theta - \mu^+ \rangle \tag{3.9}$$

$$= \Delta\Phi(\Delta/\sigma(\boldsymbol{\theta})) + \sigma(\boldsymbol{\theta})\phi(\Delta/\sigma(\boldsymbol{\theta})) \tag{3.10}$$

where $\langle \cdot \rangle$ denotes the expectation, Δ is defined in Eq. (3.8) and $\phi(\cdot)$ is the normal probability density function. The EI criterion is derived as the expected difference between the function value in a new location $f(\boldsymbol{\theta})$ and the current best μ^+.

Thus, EI aims at maximizing f by the biggest margin and yields optimization that is less prone to getting stuck in a local optimum. Nevertheless, using a tuning parameter $\xi > 0$ allows us to control the exploration vs exploitation trade-off (Lizotte et al., 2012). In Fig. 3.1, we provide an example of how PI and EI demonstrate different sampling behavior over time on the same objective function.

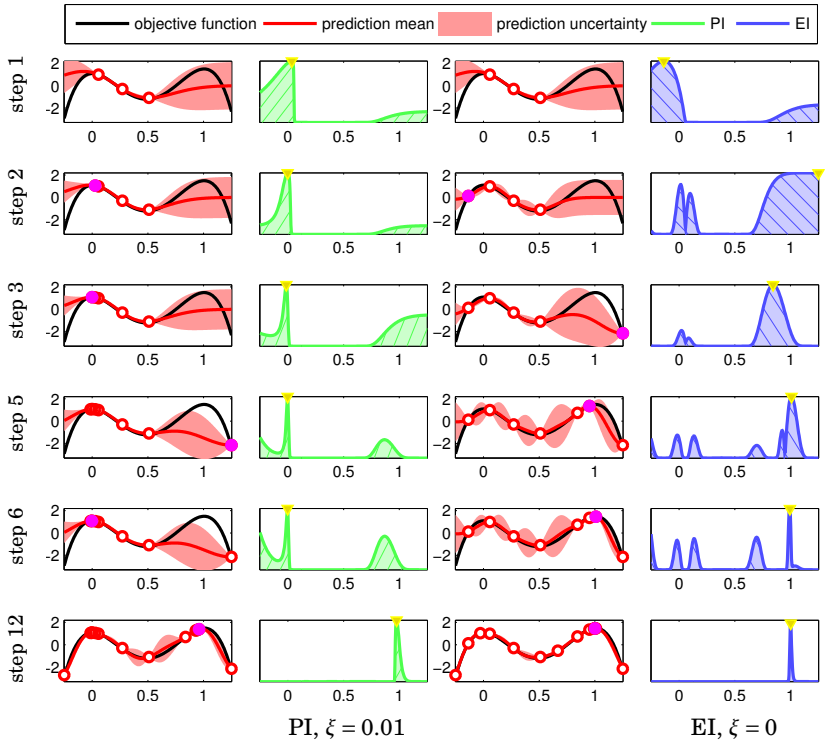

Figure 3.1. A 1-dimensional demonstration of Bayesian optimization. The objective function f is shown with the black line. The circles represent sampled values of the objective function $\{f(\boldsymbol{\theta}_i)\}_{i=1,\ldots,t}$. The red line is the prediction mean $\mu(\boldsymbol{\theta}_i)$ and pink fill color is the uncertainty $\sigma^2(\boldsymbol{\theta}_i)$ (± 2 standard deviation). The yellow star shows the new sample locations $\boldsymbol{\theta}_{t+1}$ proposed by Bayesian optimization so as to maximize the acquisition function. The magenta circle indicates the latest sampled location where the objective function is evaluated. Note that all the horizontal axes show sample locations $\boldsymbol{\theta}$. The first and second columns show the progress of the Bayesian optimization with probability of improvement acquisition function. The second column corresponds to the PI (3.7) acquisition function. The third and fourth columns show the progress of the Bayesian optimization with expected improvement acquisition function. The fourth column corresponds to the EI (3.9) acquisition function. The objective function is maximized in this example. The figure shown here is adopted from Publication III.

Figure 3.1 illustrates a 1-dimensional maximization procedure using BO. We start with three function evaluations and show the sampled points, the GP fits (with the red line) and the posterior uncertainty (with the pink filled areas). The acquisition functions are shown along the subplots of the objective function.

The new location (marked with a yellow star) is chosen so that it maximizes the acquisition function. As the optimization proceeds, we collect more samples and finally find the maximum of the objective function. One can notice that, compared to PI, EI favors exploration as it samples from regions with higher uncertainty. The objective function approximation obtained with EI improves much faster compared to PI. EI is also able to find the global maximum earlier. In this case, one could argue that using a larger value of ξ with PI could result in more exploration and faster optimization. However, selecting a good value for ξ would certainly require more computational time.

3.2 Filter Likelihood method

In tuning the chaotic systems, the likelihood is formulated for a complex system represented as a state-space model. The likelihood computations can be done with the filtering techniques (Hakkarainen et al., 2012). Note that the likelihood from the filtering techniques is the objective function f.

Filtering methods evaluate the likelihood by sequentially estimating the dynamically changing model state $\mathbf{s}_k$ for a given observation sequence $\mathbf{y}_{1:k} = \{\mathbf{y}_1, \ldots, \mathbf{y}_k\}$. Filters work by iterating two steps: prediction and update. In the prediction step, the current distribution of the state is evolved with the dynamical model to the next time step. The predictive distribution is given by the integral

$$p(\mathbf{s}_k|\mathbf{y}_{1:k-1}, \boldsymbol{\theta}) = \int p(\mathbf{s}_k|\mathbf{s}_{k-1}, \boldsymbol{\theta}) p(\mathbf{s}_{k-1}|\mathbf{y}_{1:k-1}, \boldsymbol{\theta}) d\mathbf{s}_{k-1}. \tag{3.11}$$

As this integral generally does not have a closed form solution, it is usually approximated in one way or another. This yields different filtering techniques such as extended Kalman filter, ensemble Kalman filter, particle filter and so on.

The state distribution is updated using a new observation $\mathbf{y}_k$ with the help of Bayes rule:

$$p(\mathbf{s}_k|\mathbf{y}_{1:k}, \boldsymbol{\theta}) \propto p(\mathbf{y}_k|\mathbf{s}_k, \boldsymbol{\theta}) p(\mathbf{s}_k|\mathbf{y}_{1:k-1}, \boldsymbol{\theta}). \tag{3.12}$$

This posterior is used inside the integral Eq. (3.11) to obtain the prior for the next time step.

The likelihood $p(\mathbf{y}_{1:n}|\boldsymbol{\theta})$ of the model parameters can be computed from the quantities evaluated in the filtering procedure:

$$p(\mathbf{y}_{1:K}|\boldsymbol{\theta}) = p(\mathbf{y}_1|\boldsymbol{\theta}) \prod_{k=2}^{K} p(\mathbf{y}_k|\mathbf{y}_{1:k-1}, \boldsymbol{\theta}), \tag{3.13}$$

where $p(\mathbf{y}_k|\mathbf{y}_{1:k-1}, \boldsymbol{\theta})$ is calculated based on the marginal posterior of the states:

$$p(\mathbf{y}_k|\mathbf{y}_{1:k-1}, \boldsymbol{\theta}) = \int p(\mathbf{y}_k|\mathbf{s}_k, \boldsymbol{\theta}) p(\mathbf{s}_k|\mathbf{y}_{1:k-1}, \boldsymbol{\theta}) d\mathbf{s}_k.$$

The goal in our work was to search for parameters that maximize the likelihood $f(\boldsymbol{\theta}) = p(\mathbf{y}_{1:K}|\boldsymbol{\theta})$ in Eq. (3.13).

Extended Kalman filter (EKF) is a filtering technique in which the integrals are approximated by linearization of the forward model $\mathscr{M}$ and the observation operator $\mathscr{K}$ around the current state estimate. Assuming that the observation error is normally distributed with zero mean and covariance matrix $\mathbf{R}_k$, the linearization yields:

$$p(\mathbf{y}_{1:n}|\boldsymbol{\theta}) \propto \exp\left(-\frac{1}{2}\sum_{k=1}^{n} \mathbf{r}_k^T \left(\mathbf{C}_k^y(\boldsymbol{\theta})\right)^{-1} \mathbf{r}_k + \log|\mathbf{C}_k^y(\boldsymbol{\theta})|\right) \tag{3.14}$$

$$\mathbf{C}_k^y(\boldsymbol{\theta}) = \mathbf{K}_k \left(\mathbf{M}_k \mathbf{C}_{k-1}^{\text{est}} \mathbf{M}_k^T + \mathbf{Q}_k(\boldsymbol{\theta})\right) \mathbf{K}_k^T + \mathbf{R}_k \tag{3.15}$$

where $\mathbf{r}_k = \mathbf{y}_k - \mathscr{K}(\mathbf{s}_k^p)$ are the prediction residuals, $\mathbf{M}_k$ and $\mathbf{K}_k$ are the linearization of $\mathscr{M}$ and $\mathscr{K}$ operators, respectively, $\mathbf{C}_{k-1}^{\text{est}}$ is the estimated covariance of $p(\mathbf{s}_k|\mathbf{y}_{1:k}, \boldsymbol{\theta})$ at time $k-1$ and $|\cdot|$ denotes the matrix determinant.

When the dimensionality of the tuned model is too large, the extended Kalman filter suffers from the memory issues. Another problem is that the linearization is often too cumbersome for highly complex models. In such scenarios, more sophisticated techniques like the stochastic ensemble Kalman filters (EnKF) are often used for filtering.

The basic idea of EnKF is that the posterior distribution of the states is approximated by using the sample statistics, which are computed using a relatively small number of ensemble members propagated by the model at every assimilation step. The stochastic filters involve random perturbations of the model states and observations, which introduces randomness in the likelihood evaluation. More details on EnKF can be found, for example, in Evensen (2007).

Interconnection between the Filter Likelihood and BO methods

The evaluation of the likelihood requires running a data assimilation process for a computationally expensive model. Thus, each function evaluation is expensive and there can be noise in the likelihood, for instance, the noise caused by stochastic filtering techniques. The BO facilitates the optimization process by using a GP based model as a surrogate of the tuned chaotic system. Thus, in addition to the original tuning problem, that is the estimation of the parameters of a chaotic system, BO also requires the estimation of the parameters of the surrogate model.

The Fig. 3.2 shows an illustration of the interconnection between the BO and the Filter Likelihood methods. The data assimilation process consists of the chaotic system and the filtering technique. The objective function for BO is $f(\boldsymbol{\theta})$ that is the output (log-likelihood function values) of the system. $\boldsymbol{\theta}$ and $\mathbf{y}$ represent the input (parameters) and observations, respectively. A GP based surrogate model is learned using pairs of input-output (data). Therefore, the optimization consists of two processes: the estimation of the model parame-

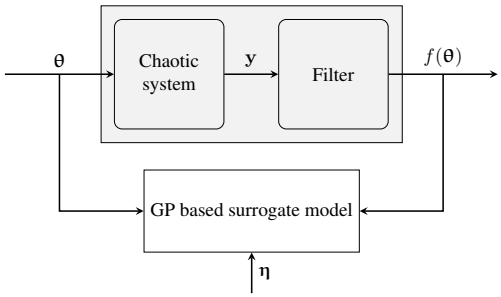

Figure 3.2. A schematic illustration of the models tuned using BO. The chaotic system and filtering technique combined together represent a data assimilation process. $\boldsymbol{\theta}$ is the input (parameters), $\mathbf{y}$ are observations and $f(\boldsymbol{\theta})$ is the output (log-likelihood function values) of the system. Pairs of input-output (data) are used to learn a GP based surrogate model. There are two estimation processes at play during BO: first is the estimation of the model parameters with the Filter likelihood method. Second is a *meta-estimation* process which optimizes the GP based surrogate model. $\boldsymbol{\eta}$ represents hyperparameters of the GP model.

ters using the Filter Likelihood method and a *meta-estimation* process which optimizes the GP based surrogate model.

3.3 Experimental results

The study of the calibration of the error covariance matrix in Publication I uses the two-layer quasi-geostrophic (QG) model where the parameterization of the covariance matrix consists of four tunable parameters. In Publication II, the parameterization introduced for the covariance matrix assures its positive definiteness for any combination of the four tunable parameters, which is important for the stability of BO. In Publication III, we have performed two studies: first, we consider parametric tuning of a simplified atmospheric model with a noiseless objective function. The tuned model is the two-layer QG model with four tuned parameters that define the model error covariance matrix of the corresponding data assimilation system. Second, we consider the parametric tuning of a chaotic system with a noisy likelihood function. We use the parameterized Lorenz 95 model as a test model, similarly to previous studies. The goal is to explore the suitability of the BO methodology for parametric tuning of full scale climate and weather models. The likelihood formulation is more relevant for (but not limited to) parametric tuning of NWP systems, as it is essentially built around the accuracy of short-term forecasts.

Parametric tuning of the QG model

A detailed description of the two-layer quasi-geostrophic (QG) model is given in section 3.1 of Publication I. The figure 5.1 (left-panel) in Publication I shows a scatter plot of the cost function values versus RMSE values using different

values for the meta-parameters of the model error covariance matrix in case of the QG model. The figure 5.2 (right-panel) in Publication I shows the forecast skill of the QG model with different values for the meta-parameters of the model error covariance matrix. From these and other results in Publication I it can be concluded that model error covariance matrix is an important tuning factor and the fact that finding the optimal meta-parameters using an approach that requires fewer evaluations of the objective function in expensive-to-evaluate models was established.

In the two-layer QG model, the meta-parameters were four variables that constructed the model error covariance matrix. In Publication III, the figure 4 illustrates the surface of the objective function, w.r.t to the four variables, estimated from BO after parametric tuning of the QG model. The figure 5 in Publication III shows the iterative process of BO which results in finding the best meta-parameters of the model error covariance matrix for the QG model.

Parametric tuning of the Lorenz 95 model

In the Lorenz 95 model, the tuning parameters were two variables that were used in the construction of a polynomial parameterization. The section 5.1 in Publication III formulates the polynomial parameterization. The comparison of the BO approach and the CMA-ES method is presented in the section 5.4 of Publication III. Figure 3.3 shows the average behavior of BO at each iteration of BO. In the Fig. 3.3, each circle is computed by

$$\mathscr{C}_t = \log\left(f(\boldsymbol{\theta}_t) - \max(f_b^*, f_c^*, f_d^*)\right), \tag{3.16}$$

where f_b^* is the maximum (best log-likelihood function value) obtained from the 200 BO experiments and f_c^* is the maximum (best log-likelihood function value) obtained from 200 runs of CMA-ES method. Each point on the plus marked line is the average value of the 200 samples of $\mathscr{C}_t$ and the solid line shows its maximum $\max(\mathscr{C}_{1:t})$: the maximum obtained upto the current iteration t. Here, $\max(f_b^*, f_c^*, f_d^*) = f_b^*$, hence, a small nugget 0.1 is added in Eq. (3.16) before taking the logarithm in order to avoid computing log(0). The dashed line is the desired log-likelihood value f_d^* which is plotted using Eq. (3.16).

In Publication III, the goodness of the optimization scheme is tested by computing the forecast accuracy. A two-dimensional grid of the parameter space was used to compute the average forecast error for different parameter values. In Fig. 3.4(a), the relationship between the average forecast error and the tuned parameters is shown where the contour lines represent the forecast error, the mathematical description of the forecast error is given in the Publication III. The magenta circles indicate the parameter values used to initialize the BO method. The black circles show the locations corresponding to samples obtained from BO. The red cross indicates location of the maximum obtained using BO.

Similarly, the relationship between the EnKF likelihood function and the tuned parameters is shown in Fig. 3.4(b). The contour lines represent the EnKF

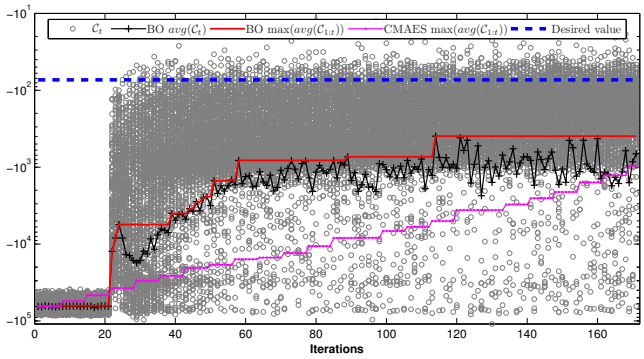

Figure 3.3. Parametric tuning of the Lorenz 95 model. The solid (red) and the dot marked (magenta) lines illustrate the average behavior of BO and CMA-ES, respectively. A total of 200 experiments are performed using each method with random initializations of the initial parameter values in the same region in each experiment. This results in 200×171 likelihood function evaluations for each method. Each gray color circle $\mathscr{C}_t$ is computed using (3.16). Each point on the plus marked (black) line is the average value of the 200 samples of $\mathscr{C}_t$ and the solid (red) line shows its maximum $\max(\mathscr{C}_{1:t})$: the maximum obtained upto the current iteration t, for BO. The dot marked (magenta) line shows the same for CMA-ES method.

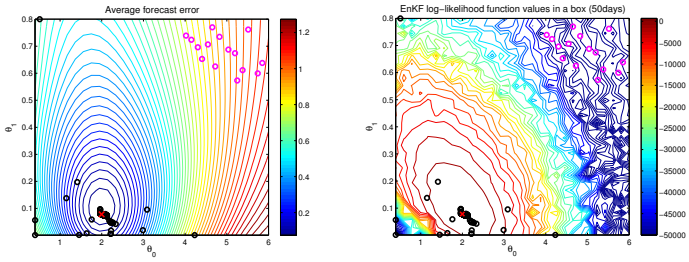

Figure 3.4. (a) The contour lines represent the average forecast error. The contours are drawn using the error values computed on an evenly spaced two-dimensional grid. The magenta circles show the parameter values used to initialize the BO method. The black circles show the locations corresponding to samples obtained from BO. The red cross indicates location of the maximum obtained using BO. The simulation length of the model used for the BO experiment was 50 days. (b) The contour lines represent the EnKF log-likelihood function values obtained using a 50 days simulation length of the model. The contours are drawn using the values computed on the same two-dimensional grid as in (a). Note that the contour lines are curly due to noisy values obtained with the EnKF likelihood function. The circles and cross mark correspond to the same result as shown in (a) on the left.

log-likelihood function values obtained using a 50 days simulation length of the model. The contours are drawn using the values computed on the same two-dimensional grid as in Fig. 3.4(a). Note that the contour lines are curly due to noisy values obtained with the EnKF likelihood function. The circles and cross mark correspond to the same result as shown in Fig. 3.4(a).

From these results and others shown in the Publication III, it can be concluded that the BO method is a suitable tool for optimizing the meta-parameters in complex climate and numerical weather prediction models.

4. Adversarial training of neural network models

A plethora of techniques for regularization have been introduced, especially, for neural networks such as early stopping of training the parameters (see Bishop, 2006, sec. 5.5), data augmentation, i.e. the training with transformed data or noise injection to inputs, weights and activations (Sietsma and Dow, 1991; Vincent et al., 2008), restrictions on the parameter values (Poggio and Girosi, 1990; Rifai et al., 2011), and ensemble learning methods that combine the outputs of different hypotheses. Similar to data augmentation is the adversarial training method that has recently become a popular method for deep neural network regularization (Kurakin et al., 2016; Salimans et al., 2016; Makhzani et al., 2015). In adversarial training, inputs close to the original inputs are found so that the model is trained to generate the same output on these as on the original inputs (Goodfellow et al., 2016). Such inputs which are called adversarial examples are often extremely similar to the original examples, so much that a human observer is unable to tell the difference. In Szegedy et al. (2014); Goodfellow et al. (2014b), the authors have shown such images.

First, Goodfellow et al. (2014b) used adversarial training as a regularization method to improve the results on the MNIST benchmark. Later, a more interesting strategy called virtual adversarial training (VAT) was introduced by Miyato et al. (2016). One of the main difference between VAT and adversarial training is that the later is a supervised training method. VAT works for both the unsupervised and semi-supervised settings and it utilizes soft labels obtained from the neural network output in the unsupervised setting. In case some of the true labels are available it works as a semi-supervised training method. It has turned out that VAT outperformed many recent regularization methods except the state-of-art Ladder Networks (Rasmus et al., 2015).

In this work, an alternative loss function is used for adversarial training. The method is useful for semi-supervised training since unlabeled data is sufficient for the computation of the regularization penalty. The approach can easily combine with the Ladder Networks by simply utilizing the structure of the Ladder Networks. The theoretical connections of the regularization framework were presented in Publication IV. The experimental results of the adversarial variant of the regularization framework on the MNIST benchmark are presented

in this thesis.

Section 4.1 explains the adversarial training method. In section 4.2, the regularized autoencoders, especially, the modifications of the canonical autoencoder, namely, the denoising autoencoder and the contractive autoencoder are presented. These modifications enable the canonical autoencoders to perform regularization. The Ladder Networks are discussed in section 4.2.3. The connections of the regularization framework and other recent regularization methods have been discussed in section 4.3. Section 4.5 presents the results of the experiments performed with the MNIST data.

4.1 Adversarial training

A simple method to increase the generalization ability of a machine learning model is to train it with more data. However, this is usually possible up to a certain limit, especially, in the case of labeled data. A possible solution to this problem is generation of training samples using prior knowledge. For example, in object recognition the image dataset size can be increased by image transformations: rotation, scale and shift. Such regularization methods are broadly categorized as dataset augmentation.

The method of injecting noise in the inputs of a neural network (Sietsma and Dow, 1991; Vincent et al., 2008) is another example of dataset augmentation. Similarly, adversarial training utilizes additional data in the form of adversarial examples (Szegedy et al., 2014). In dataset augmentation, the transformations are manually specified which incurs the model to be invariant to the specified directions of change in the input. This is different for adversarial training that requires the model to be invariant to all directions of change in the input as long as such changes are small (Goodfellow et al., 2014b).

In the seminal work of Szegedy et al. (2014) and later in Goodfellow et al. (2014b), it was shown that addition of small perturbations to training examples resulted in the misclassification of those training examples with high confidence by the model. In this case, the adversarial examples were generated by perturbations via random noise in the input space in the directions to which the label probability of the model is most sensitive to. Goodfellow et al. (2014b) developed the adversarial training method that successfully improves the generalization performance while the model remains robust to adversarial perturbations.

Let us denote $\mathbf{x}$ as the input, t as the associated labels, and $\boldsymbol{\theta}$ as the model parameters. In case, the method is applied to a classifier, adversarial training adds the following penalty to the training criteria as follows

$$\mathscr{R}_{\text{adv}}(t,\mathbf{x},\boldsymbol{\theta}) = -\log\, p(t|\mathbf{x}+\bar{\mathbf{x}}_{\text{adv}};\boldsymbol{\theta}) \tag{4.1}$$

$$\text{where } \bar{\mathbf{x}}_{\text{adv}} = \min_{\bar{\mathbf{x}};\|\bar{\mathbf{x}}\|\leq\epsilon} \log\, p(t|\mathbf{x}+\bar{\mathbf{x}};\hat{\boldsymbol{\theta}}) \tag{4.2}$$

where $\bar{\mathbf{x}}$ is a perturbation on the input and a constant set of the current parameters $\hat{\boldsymbol{\theta}}$ are used. The reason for the use of $\hat{\boldsymbol{\theta}}$ is to highlight that the back-

propagation algorithm is not used during the adversarial example generation process. During training, the current model $p(t|\mathbf{x};\hat{\boldsymbol{\theta}})$ is trained to be robust against worst case perturbations $\bar{\mathbf{x}}_{\text{adv}}$ through minimization of the Eq. 4.2 w.r.t $\boldsymbol{\theta}$. However, minimization of this exactly is in general not possible because the exact minimization w.r.t $\bar{\mathbf{x}}$ is intractable in many cases, especially, for neural networks. In Goodfellow et al. (2014b), it was proposed to approximate this value by linearization of the training criteria $\log p(t|\mathbf{x};\hat{\boldsymbol{\theta}})$ around $\mathbf{x}$. In case of L_2 norm with the linear approximation the resulting adversarial perturbation is generated by

$$\bar{\mathbf{x}}_{\text{adv.}} \approx -\epsilon \frac{\mathbf{g}}{\|\mathbf{g}\|_2} \text{ where } \mathbf{g} = \nabla_{\mathbf{x}} \log p(t|\mathbf{x};\hat{\boldsymbol{\theta}}) \,, \tag{4.3}$$

where ϵ is a positive scalar meta-parameter and in case of L_∞ norm (Goodfellow et al., 2014b) the resulting adversarial perturbation reads as follows

$$\bar{\mathbf{x}}_{\text{adv.}} \approx \epsilon \, \text{sign}(\mathbf{g}) \,, \tag{4.4}$$

where $\mathbf{g}$ is the same as in Eq. 4.3. This way of adversarial example generation is called the *fast gradient sign method*, as mentioned in Goodfellow et al. (2014b). Note that this gradient can be efficiently computed for neural networks by backpropagation. The authors in Goodfellow et al. (2014b) showed better generalization performance using this method as compared to random perturbations.

Virtual adversarial training (Miyato et al., 2016) is closely related to adversarial training. In VAT, virtual adversarial examples are generated by maximizing the Kullback-Leibler divergence between encoding distributions for the example $\mathbf{x}$ and its perturbed version $\bar{\mathbf{x}}_{\text{vadv}}$. The additional penalty introduced by the VAT method is given as

$$\mathscr{R}_{\text{vadv.}}(\mathbf{x},\boldsymbol{\theta}) = \text{KL}\left[(p(\mathbf{y}|\mathbf{x},\hat{\boldsymbol{\theta}})||p(\mathbf{y}|\mathbf{x}+\bar{\mathbf{x}}_{\text{vadv.}},\boldsymbol{\theta})\right] \tag{4.5}$$

$$\text{where } \bar{\mathbf{x}}_{\text{vadv.}} = \max_{\bar{\mathbf{x}};\|\bar{\mathbf{x}}\|\leq\epsilon} \text{KL}\left[(p(\mathbf{y}|\mathbf{x},\hat{\boldsymbol{\theta}})||p(\mathbf{y}|\mathbf{x}+\bar{\mathbf{x}},\hat{\boldsymbol{\theta}})\right] \,, \tag{4.6}$$

where $\text{KL}[\cdot]$ denotes the KL divergence between the encoding and perturbed distributions. Here the classifier is made resistant to perturbations in directions to which it is most sensitive on the current model $p(\mathbf{y}|\mathbf{x};\hat{\boldsymbol{\theta}})$. The VAT training only requires $\mathbf{x}$ and uses only soft labels $\mathbf{y}$ from the training model instead of the true labels $\mathbf{t}$. This makes it possible to use VAT for semi-supervised and unsupervised learning. In Miyato et al. (2016), the authors proposed an efficient method of approximating Eq. 4.5 using backpropagation.

4.2 Regularized autoencoders

The latest revival in deep learning neural networks started with the idea based upon using unsupervised training as a prior step to supervised training such

as training a classifier (Hinton et al., 2006; Bengio et al., 2007; Ranzato et al., 2007). This method was called unsupervised pre-training or greedy layer-wise unsupervised pre-training (Goodfellow et al., 2016), more specifically. Unsupervised pre-training was generally used for better intialization of the neural network model parameters. By using unsupervised pre-training layer-by-layer in a deep neural network would map the inputs to more useful intermediate representations. Better initialization of the model parameters was also seen as having a regularization effect (Erhan et al., 2010). More generally, the method used was based on the idea that learning about the data distribution itself through unsupervised training methods could help with tasks where an input is mapped to a different output, subsequently.

Another important realization from unsupervised pre-training was that how unlabeled data could be made useful for training tasks requiring labels? After instigation of deep learning, unsupervised training has become quite obsolete but it has motivated the development of semi-supervised training methods (Goodfellow et al., 2016). Semi-supervised training methods simultaneously use unsupervised and supervised training. Unsupervised pre-training also motivated research on specialized autoencoders. The original autoencoders were single layer models mainly used for tasks such as dimensionality reduction and feature learning. With the success of unsupervised pre-training the focus shifted towards multi-layer (stacked) autoencoders (Vincent et al., 2010) which could be used for representation learning (Goodfellow et al., 2016).

Two modifications to the standard autoencoder have shown to regularize the autoencoder model, namely the denoising autoencoder and the contractive autoencoder. In the following sections, a brief introduction is given for the denoising autoencoder and the contractive autoencoder.

4.2.1 Denoising autoencoder

The denoising autoencoder (DAE) is trained to reconstruct the clean input from the corrupted version of it (Vincent et al., 2008). The DAE encoder maps the corrupted inputs $\tilde{\mathbf{x}}$ to a hidden representation $\tilde{\mathbf{h}} = f(\tilde{\mathbf{x}})$ and the decoder maps $\tilde{\mathbf{h}}$ to the output $\hat{\mathbf{x}} = g(\tilde{\mathbf{h}})$. Typically, the corruption can be of two forms: an additive isotropic Gaussian noise $\tilde{\mathbf{x}} = \mathbf{x} + \mathcal{N}(0, \sigma^2 I)$ and a binary masking of the percentage v of the inputs values randomly set to 0. The amount of corruption σ or v determines the amount of regularization.

The DAE training criteria automatically avoids learning the identity mapping between the inputs and outputs of the model. The standard autoencoders minimize some loss function $\mathscr{L}(\mathbf{x}, g(f(\mathbf{x})))$, where the dissimilarity between $g(f(\mathbf{x}))$ and $\mathbf{x}$ is penalized, for example, the L^2 norm. However, it is likely to learn only the identity function given sufficient capacity in the model. The DAE instead minimizes a loss function $\mathscr{L}(\mathbf{x}, g(f(\tilde{\mathbf{x}})))$, where $\tilde{\mathbf{x}}$ is the corrupted version of $\mathbf{x}$. Therefore, the DAE learns to perform denoising rather than learning of the identity mapping. This denoising of the input helps to learn the structure of the

input distribution that is beneficial for representation learning (Vincent et al., 2010).

4.2.2 Contractive autoencoder

In the contractive autoencoder (CAE), an additional regularizer term is introduced that penalizes the derivatives of the hidden representation $\mathbf{h} = f(\mathbf{x})$ with respect to the input (Rifai et al., 2011). The penalty is computed from the sum of the squared elements of the Jacobian matrix of partial derivatives, known as the Frobenius norm,

$$\mathscr{L}(\mathbf{x}, g(f(\mathbf{x}))) + \Omega(\mathbf{h}), \tag{4.7}$$

$$\text{where } \Omega(\mathbf{h}) = \lambda \left\| \frac{(\partial f(\mathbf{x}))}{\partial \mathbf{x}} \right\|^2. \tag{4.8}$$

λ controls the amount of regularization of the model.

The idea of the CAE is to construct robust features or hidden representations i.e., to make the CAE less sensitive to small perturbations close to the input examples. The name contractive is given by the manifold perspective that is the CAE encourages the near by points in the input space to be mapped to near by points in the hidden space. This can be seen as the contraction of the input neighborhood to a smaller neighborhood in the output. See (Goodfellow et al., 2016, chp. 14), for a detailed discussion on CAE from the manifold perspective.

4.2.3 Ladder Networks

The Ladder Network is a variant of the autoencoder, proposed by Valpola (2015). The structure of the Ladder Network consists of horizontal lateral connections at regular intervals between the encoder and the decoder, hence the name Ladder Network. The purpose of the lateral (skip) connections is to relieve the pressure of local feature extraction from the higher layers of the model. This enables the higher layers to focus on abstract invariant features Valpola (2015). Although the training of the Ladder Network is similar to the denoising autoencoder the corrupted representations are denoised at every level of the model.

The Ladder Network has been used as an efficient method for semi-supervised training and it has achieved state-of-the-art results on benchmark datasets (see e.g., Rasmus et al., 2015). The Fig. 4.1 shows the structure of the Ladder Network that consists of the two encoder paths: a corrupted path that is attached to a decoder path and a clean path. The Ladder Network shown in the Fig. 4.1 is a simplified depiction of the one presented by Rasmus et al. (2015). This structure allows for the study of the regularization framework presented in Publication IV and the adversarial variant of it in section 4.4. In the section 4.5, the results of the combination of the Ladder Network with the regularization framework proposed in Publication IV are shown.

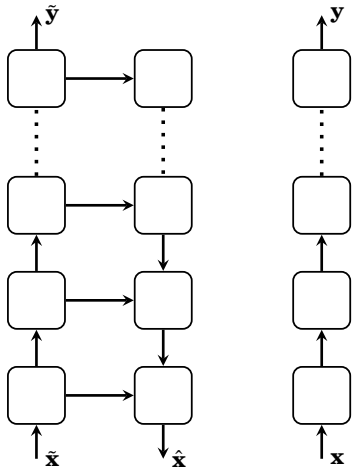

Figure 4.1. A simplified depiction of the Ladder Network framework. Each rounded square represents a neural network layer. There are two encoder paths: a clean path (with $\mathbf{x}$ as input and $\mathbf{y}$ as output) and a corrupted path (with $\tilde{\mathbf{x}}$ as input and $\tilde{\mathbf{y}}$ as output) that is connected to the decoder path (with $\hat{\mathbf{x}}$ as the reconstructions). Noise can be added to the input, hidden and output layers of the corrupted path.

4.3 Regularization connections

In Publication IV, a regularization framework was presented and its connections to other related regularization methods were discussed. The regularization framework consisted of two identical mapping functions where the clean input was mapped to $f(\mathbf{x})$ and the corrupted input was mapped to $f(\tilde{\mathbf{x}})$. The regularization term $\|f(\mathbf{x}) - f(\tilde{\mathbf{x}})\|^2$ was computed as the squared Euclidean distance between the two mapping functions. The regularization framework in Publication IV can be seen as part of the Ladder Network structure shown in Fig. 4.1.

It was shown in Publication IV that the presented regularization framework was an instance of the Pseudo-Ensemble Agreement regularization approach by Bachman et al. (2014), it generalized the noise regularization (Sietsma and Dow, 1991), and it was a simplification of the regularization term by the Ladder Networks in Rasmus et al. (2015). The connection to the virtual adversarial training (VAT) (Miyato et al., 2016) was shown and VAT could be interpreted as penalizing the largest eigenvalue of a Fisher information matrix.

A central method connecting the different regularization methods in Publication IV was the penalization of the Jacobian. Poggio and Girosi (1990) proposed to penalize a neural network by the Frobenius norm of the Jacobian. This method of regularization was more recently used for regularization of the autoencoder model by Rifai et al. (2011) as discussed in the section 4.2.2.

4.4 Adversarial training for Ladder Networks

In this section, the adversarial variant of the regularization framework is presented. Let us say a dataset $\mathbf{X}$ consists of N samples $\{\mathbf{x}^{(n)}\}_{n=1}^{N}$ and M labels $\{\mathbf{t}^{(m)}\}_{m=1}^{M}$. In supervised training $M = N$ while in semi-supervised training $M << N$. In our case, the mapping functions are feed-forward neural networks. We can use the semi-supervised Ladder Network by Rasmus et al. (2015) for this case. The clean encoder path gives the output $\mathbf{y} = f(\mathbf{x})$ while the corrupted encoder path gives the output $\tilde{\mathbf{y}} = f(\tilde{\mathbf{x}})$.

The training is done by the minimization of the training criterion $\mathscr{L}(\mathbf{t}, \mathbf{X}, \boldsymbol{\theta}) = \mathbb{E}_{p(\mathbf{t}|\mathbf{x})}(\cdot)$, iteratively, where the expectation $\mathbb{E}_{p(\mathbf{t}|\mathbf{x})}$ is taken over the training data set $\mathbf{X}$. A regularization term $\mathscr{R}_{\text{cadv.}}$ is added to the training criterion in order to help it generalizing better to unseen data. Hence, the total loss function minimized during training, reads as follows

$$\alpha \mathscr{L}(\mathbf{t}, \mathbf{X}, \boldsymbol{\theta}) + \beta \mathscr{R}_{\text{cadv.}}, \tag{4.9}$$

where α and β are positive scalar hyperparameters.

The proposed method is similar to adversarial training (Goodfellow et al., 2014b) but adds a different regularization to the training criteria. The regularization penalty in this case is given by

$$\mathscr{R}_{\text{cadv.}}(\mathbf{x}, \boldsymbol{\theta}) = \mathbb{E}_{p(\mathbf{x}, \tilde{\mathbf{x}}_{\text{cadv.}})} \| f(\mathbf{x}; \hat{\boldsymbol{\theta}}) - f(\mathbf{x} + \tilde{\mathbf{x}}_{\text{cadv.}}; \boldsymbol{\theta}) \|^2 \tag{4.10}$$

$$\text{where } \tilde{\mathbf{x}}_{\text{cadv.}} = \max_{\tilde{\mathbf{x}}: \|\tilde{\mathbf{x}}\| \leq \epsilon} \| f(\mathbf{x}; \hat{\boldsymbol{\theta}}) - f(\mathbf{x} + \tilde{\mathbf{x}}; \hat{\boldsymbol{\theta}}) \|^2 \tag{4.11}$$

where $\mathbb{E}_{p(\mathbf{x}, \tilde{\mathbf{x}}_{\text{cadv.}})}$ is the expectation taken over training data set $\mathbf{X}$. Note that the constant set of the current parameters $\hat{\boldsymbol{\theta}}$ are used and that the backpropagation algorithm is not used during the adversarial example generation process. We use the approximation of Eq. 4.11 given as follows

$$\mathbf{x}_{\text{adv.}} \approx \epsilon \ \text{sign}(\mathbf{g}) \ \text{where } \mathbf{g} = \nabla_{\mathbf{x}} \| \mathbf{y} - \tilde{\mathbf{y}} \|^2 \tag{4.12}$$

where ϵ is a positive scalar meta-parameter, $\mathbf{y} = f(\mathbf{x}; \hat{\boldsymbol{\theta}})$ denotes the network output given a sample $\mathbf{x}$, $\tilde{\mathbf{y}} = f(\tilde{\mathbf{x}}; \hat{\boldsymbol{\theta}}')$ denotes the network output given a corrupted version of it, and the corrupted sample $\tilde{\mathbf{x}}$ is obtained by adding white Gaussian noise to the clean sample so that $\tilde{\mathbf{x}} \sim \mathscr{N}(\mathbf{0}, \eta^2 \mathrm{I})$, with I denoting the identity-matrix and η is a positive scalar that is used to control the amount of corruption.

In Publication IV, it was shown how the penalization of the Jacobian is a central method connecting the proposed regularization to other methods. It was shown that there exists a connection between the proposed regularization and the contractive autoencoder by Rifai et al. (2011). Therefore, the method presented in this section is called the *contractive adversarial training* (CAT). This method is easily implemented in the Ladder framework where we obtain $\mathbf{y}$ and $\tilde{\mathbf{y}}$ from the clean and corrupted encoder paths, respectively. Similar to VAT,

this method only requires $\mathbf{x}$ and uses only soft labels $\mathbf{y}$ from the training model instead of the true labels $\mathbf{t}$. This makes it suitable for semi-supervised and unsupervised learning. In contrast to VAT where a complex method is required for approximation of the virtual adversarial example, here the process simply requires to perform the linearization with the sign applied to it.

4.5 Experimental results

We have tested the proposed regularization framework on the MNIST benchmark dataset. The standard machine learning task for MNIST data is the classification of handwritten digits between 0 and 9. The dataset consists of 60,000 labeled examples for training and 10,000 labeled examples for testing. The training data is shuffled with a random seed for each training epoch. This is also called the permutation invariant training method that has been previously used for MNIST, especially, in the work by Rasmus et al. (2015). MNIST has been extensively used in machine learning research, especially, for deep learning.

In the experiments, we perform both supervised and semi-supervised learning. In case of supervised training we used all the labels and a few labels are used for semi-supervised training. The aim of the experiments was to test the proposed framework as a regularizer, combine the regularizer to the Ladder network model in Rasmus et al. (2015) and compare the classification results with other regularization methods.

The baseline model was a feed-forward neural network (MLP) with a 5 layer structure with layer sizes set to be 784-1000-500-250-250-250-10. This structure was chosen so that the model can be easily combined and tested against the Ladder model in Rasmus et al. (2015) that used the same structure. This structure has the property of being deep enough to test the scalability while not being an overkill for MNIST task (Rasmus et al., 2015). The Table 4.1 consists of the test error percentages of different regularization methods on the MNIST data with different number of labeled samples. The comparison is shown with 100, 1000 (semi-supervised) and all labeled (supervised) samples.

First, we test the contractive adversarial training (CAT) by combining it to the baseline model. From Table 4.1, we observe that CAT produces better results as compared to the baseline itself for both supervised and semi-supervised settings. This shows that CAT is able to perform well as a standalone regularizer. However, these results are only better than the PEA (Bachman et al., 2014) and not as good as the other regularization methods shown in the Table 4.1.

Secondly, the CAT is combined with the Ladder network presented in Rasmus et al. (2015). As shown in 4.1, the results for the supervised learning (all labels) are very close to VAT (Miyato et al., 2016) and the Ladder Networks (Rasmus et al., 2015). For semi-supervised learning (fewer labels), the results are very close to Ladder Networks (Rasmus et al., 2015) and slightly better than the other regularization methods in the Table 4.1.

	test error percentage		
Methods with number of labels	100	1000	All
Pseudo Ensemble (PEA) (Bachman et al., 2014)	5.21	2.64	1.08
Adversarial training (Goodfellow et al., 2014b)			0.78
Virtual adversarial (Miyato et al., 2016)	2.33	1.36	0.64
Ladder Networks (Rasmus et al., 2015)	1.06 ± 0.37	0.84 ± 0.08	0.57 ± 0.02
CatGAN (Springenberg, 2015)	1.91 ± 0.1	1.73 ± 0.18	0.91
Baseline Neural Network	26.4 ± 1.85	8.7 ± 0.32	1.18 ± 0.07
Contractive adversarial (CAT)	3.05 ± 1.6	1.35 ± 0.08	0.82 ± 0.05
CAT + Ladder network	0.96 ± 0.07	0.79 ± 0.05	0.56 ± 0.05

Table 4.1. The comparison of the regularization framework with other test error results on the permutation invariant MNIST case. We show the results with 100, 1000 (semi-supervised setting) and all labeled (supervised setting) samples.

The training settings used for the experiments were similar to the settings in Rasmus et al. (2015). The training was done with a mini-batch size fixed to 100 samples in the baseline, CAT and CAT plus Ladder results. The Adam optimization algorithm (Kingma and Ba, 2015) was used for the weight updates. The initial learning rate was 0.002 for 100 training steps (epochs), followed by an annealing phase during which the learning rate was linearly reduced to zero.

For CAT, we optimized the meta-parameters ϵ and η on the standard training set of $60,000$ samples that was randomly split into a $10,000$ samples validation set and $50,000$ samples training set. Each training was repeated 10 times by varying the random seed that was used for the splits. The search grid used for ϵ was $\{0.1, 0.2, 0.3, 0.4, 0.5, 1.0, 2.0, 3.0\}$ and for η it was $\{0.1, 0.2, 0.3, 0.4\}$. We found that the good value for ϵ in case of 1000 labeled samples and all labeled samples was $\epsilon = 0.2$ while in case of 100 labeled samples $\epsilon = 0.1$ was better. We found $\eta = 0.3$ to be a good value for both semi-supervised cases and the supervised case. These meta-parameter values were also used when CAT was combined with the Ladder network. We adopted the same meta-parameters for the Ladder Networks as presented in Rasmus et al. (2015). When CAT was applied to the Ladder network the cost multipliers were set to $\alpha = 1$ and $\beta = 1$.

For the final results all the $60,000$ training samples were used during training phase while the standard $10,000$ test samples were kept as a held-out test set. The average values for the results shown in Table 4.1 were obtained with 10 different random initializations of the weight matrices and data splits. This procedure was also the same as used by Rasmus et al. (2015). Overall the CAT method is an alternative to the adversarial training (Goodfellow et al., 2014b) and to the VAT (Miyato et al., 2016). The main advantage gained with CAT is that it can be easily combined with the existing Ladder Networks structure that has shown to be a promising state-of-the-art method for semi-supervised learning.

5. Summary and Conclusions

In this dissertation, training methods for climate and neural network models were studied. The results in Publication I revealed the importance of tuning the model error covariance matrix in EKF. The effect of calibrating the model error covariance matrix parameters on the forecast error was also studied. It was observed that the Filter Likelihood method was able to estimate the model error covariance matrix parameters that resulted in the optimal likelihood function values. The conclusion or main results from the study were that the model error covariance matrix calibration depends on the quality of the observations and that the estimation approach yields a well-tuned model in terms of the accuracy of the state estimates and model predictions. It was concluded that the studied approach can be a successful method for calibration of other climate models.

The work in Publication I motivated and lead to further studies of model parameter tuning in climate and numerical weather prediction (NWP) and was carried out in Publication II. In Publication II, the suitability of the Bayesian optimization (BO) approach was studied for tuning the climate and NWP models. The BO was applied to two benchmark models: the two-layer quasi-geostrophic (QG) model and the Lorenz 95 model. For both models the parametric schemes were used that consisted of the tuning parameters. Further results from the application of the BO method on climate and NWP models were presented in Publication III. The BO approach was compared with the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) for the Lorenz 95 model. A performance metric was formulated to test the accuracy of the tuned model. The accuracy of the tuned parameters was shown in terms of the forecast error and the lead time analysis was performed. The main results were that the BO approach found the optimal meta-parameters in both of the tuning tasks with a low number of objective function evaluations. As a conclusion the BO approach was found to be a suitable method for tuning expensive-to-evaluate objective functions in climate and NWP.

A regularization framework was proposed in Publication IV and its theoretical connections to state-of-the-art regularization methods were shown. It was demonstrated that the studied approach is a stochastic estimate of penalizing the Frobenius norm of the Jacobian of the mapping parameters, that it generalizes

noise regularization, and that it is a simplification of the canonical regularization term by the Ladder Networks. The connection to virtual adversarial training (VAT) method (Miyato et al., 2016) was investigated and it was shown that the VAT can be interpreted as penalizing the largest eigenvalue of a Fisher information matrix. In conclusion, the presented mathematical results could help in understanding the relationship between different types of regularization techniques. The work in Publication IV motivated the results presented in chapter 4, where the results are presented for the adversarial variant of the regularization framework studied in Publication IV. The framework was used to regularize the Ladder Networks, see Rasmus et al. (2015), on the MNIST classification task. From this work it was concluded that the CAT method is a good alternative to the adversarial training (Goodfellow et al., 2014b) or the VAT (Miyato et al., 2016). The CAT has the advantage that it can be easily combined with Ladder Networks that have been shown as a promising state-of-the-art semi-supervised learning method.

We have studied two central problems in machine learning both of tuning the meta-parameters and regularization that have an important role while training the climate and neural network models. Further research on the BO methods can be carried out in two separate directions: improving the BO methodology itself for climate science and in application of the BO to large scale climate models such as ECHAM5 and real weather predictions data. As the question of climate change emerges as one of the most critical challenges of our time it might be useful to use such machine learning methods to answer some of the research questions in this domain. Artificial intelligence has also emerged as one of the leading fields of research that has the potential to solve many real world and societal problems. The work done in this thesis serves as another important step towards the further research on regularization schemes that could enhance deep learning or artificial intelligence frameworks. In order to summarize this thesis work it could be easily mentioned that both BO as a tool in climate and NWP, and the proposed regularization framework are two techniques that require further exploration and research.

References

Alpaydin, E. (2014). *Introduction to machine learning*. MIT press.

Bachman, P., Alsharif, O., and Precup, D. (2014). Learning with pseudo-ensembles. In *Proc., Advances in Neural Information Processing Systems (NIPS)*, pages 3365–3373. MIT Press, Cambridge, MA, USA.

Baldi, P. and Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *Proc., Advances in Neural Information Processing Systems (NIPS)*, pages 153–160, Cambridge, MA, USA. MIT Press.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford University press.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Boyle, P. (2007). *Gaussian Processes for Regression and Optimisation*. PhD thesis, Victoria University of Wellington, New Zealand.

Brochu, E., Cora, V. M., and de Freitas, N. (2010). A tutorial on Bayesian optimization of expensive cost functions with application to active user modeling and hierarchical reinforcement learning. *arXiv:1012.2599, arXiv.org, Cornell University Library, USA*.

Chapelle, O., Schölkopf, B., and Zien, A. (2006). *Semi-Supervised Learning*. MIT Press, Cambridge, MA.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern classification*. John Wiley & Sons.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P. A., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660.

Evensen, G. (2007). *Data Assimilation: The Ensemble Kalman Filter*. Springer.

Frean, M. and Boyle, P. (2008). Using Gaussian processes to optimize expensive functions. In *Proceedings of the 21st Australasian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence*, AI '08, pages 258–267. Springer.

Goodfellow, I. J., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014a). Generative adversarial nets. In *Proc., Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680, Cambridge, MA, USA. MIT Press.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014b). Explaining and harnessing adversarial examples. *arXiv:1412.6572, arXiv.org, Cornell University Library, USA*.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout networks. *arXiv:1302.4389, arXiv.org, Cornell University Library, USA*.

Hakkarainen, J., Ilin, A., Solonen, A., Laine, M., Haario, H., Tamminen, J., Oja, E., and Järvinen, H. (2012). On closure parameter estimation in chaotic systems. *Nonlin. Processes Geophys.*, 19(1):127–143.

Hakkarainen, J., Solonen, A., Ilin, A., Susiluoto, J., Laine, M., Haario, H., and Järvinen, H. (2013). A dilemma of the uniqueness of weather and climate model closure parameters. *Tellus A*, 65:20147.

Hansen, N. and Ostermeier, A. (1996). Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Evolutionary Computation, Proceedings of IEEE International Conference on*, pages 312–317. Morgan Kaufmann.

Hauser, T., Keats, A., and Tarasov, L. (2012). Artificial neural network assisted Bayesian calibration of climate models. *Clim. Dynam.*, 39(1):137–154.

Hinton, G. E. (2012). A practical guide to training restricted Boltzmann machines. In *Neural networks: Tricks of the trade*, pages 599–619. Springer.

Hinton, G. E., Osindero, S., and Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

Hsu, C. W., Chang, C. C., and Lin, C. J. (2003). A practical guide to support vector classification.

Hutter, F., Hoos, H., and Leyton-Brown, K. (2013). Bayesian optimization with censored response data. *arXiv:1310.1947, arXiv.org, Cornell University Library, USA*.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *In: Coello C.A.C. (eds) Learning and Intelligent Optimization. LION 2011. Lecture Notes in Computer Science*, volume 6683. Springer.

Jackson, C., Sen, M. K., and Stoffa, P. L. (2004). An efficient stochastic Bayesian approach to optimal parameter and uncertainty estimation for climate model predictions. *Journal of Climate*, 17(14):2828–2841.

Järvinen, H., Räisänen, P., Laine, M., Tamminen, J., Ilin, A., Oja, E., Solonen, A., and Haario, H. (2010). Estimation of ECHAM5 climate model closure parameters with adaptive MCMC. *Atmos. Chem. Phys.*, 10(20):9993–10002.

Jones, D. R. (2001). A taxonomy of global optimization methods based on response surfaces. *J. Global Optim.*, 21(4):345–383.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Kurakin, A., Goodfellow, I. J., and Bengio, S. (2016). Adversarial machine learning at scale. *arXiv:1611.01236, arXiv.org, Cornell University Library, USA*.

Kushner, H. J. (1964). A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *ASME. J. Basic Eng.*, 86(1):97–106.

LeCun, Y., Bengio, Y., and Hinton, G. E. (2015). Deep learning. *Nature*, 521(7553):436–444.

Leutbecher, M. and Palmer, T. (2008). Ensemble forecasting. *J. Comput. Phys.*, 227(7):3515–3539.

Lizotte, D. J., Greiner, R., and Schuurmans, D. (2012). An experimental methodology for response surface optimization methods. *J. Global Optim.*, 53(4):699–736.

Lorenz, E. N. (1995). Predictability: a problem partly solved. In *Proceedings of the Seminar on Predicability*, pages 1–18. ECMWF.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I. J., and Frey, B. (2015). Adversarial autoencoders. *arXiv:1511.05644, arXiv.org, Cornell University Library, USA*.

Miyato, T., Maeda, S., Koyama, M., Nakae, K., and Ishii, S. (2016). Distributional smoothing with virtual adversarial training. In *International Conference on Learning Representations (ICLR)*.

Mockus, J. (1989). *Bayesian Approach to Global Optimization: Theory and Applications*. Kluwer Academic.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*.

Neelin, J. D., Bracco, A., Luo, H., McWilliams, J. C., and Meyerson, J. E. (2010). Considerations for parameter optimization and sensitivity in climate models. *Proceedings of the National Academy of Sciences*, 107(50):21349–21354.

Osborne, M. A., Roman, G., and Roberts, S. J. (2009). Gaussian processes for global optimization. In *3rd International Conference on Learning and Intelligent Optimization*, pages 1–15.

Pedlosky, J. (1987). *Geophysical Fluid Dynamics*. second ed. Springer-Verlag, New York.

Poggio, T. and Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497.

Ranzato, M., Poultney, C., Chopra, S., and LeCun, Y. (2007). Efficient learning of sparse representations with an energy-based model. In *Proc., Advances in Neural Information Processing Systems (NIPS)*, pages 1137–1144, Cambridge, MA, USA. MIT Press.

Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. (2015). Semi-supervised learning with Ladder Networks. In *Proc., Advances in Neural Information Processing Systems (NIPS)*, pages 3532–3540. MIT Press, Cambridge, MA, USA.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge MA, USA.

Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 833–840.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323((Oct)):533–536.

Salimans, T., Goodfellow, I. J., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training GANs. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 2234–2242. Curran Associates Inc.

Schirber, S., Klocke, D., Pincus, R., Quaas, J., and Anderson, J. L. (2013). Parameter estimation using data assimilation in an atmospheric general circulation model: from a perfect toward the real world. *Journal of Advances in Modeling Earth Systems*, 5(1):58–70.

Sietsma, J. and Dow, R. J. F. (1991). Creating artificial neural networks that generalize. *Neural Networks*, 4(1):67–79.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. *arXiv:1206.2944, arXiv.org, Cornell University Library, USA*.

Solonen, A. and Järvinen, H. (2013). An approach for tuning ensemble prediction systems. *Tellus A*, 65:20594.

Springenberg, J. T. (2015). Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv:1511.06390, arXiv.org, Cornell University Library, USA*.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. E. (2013). On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1139–1147.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. (2014). Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*.

Urban, N. M. and Fricker, T. E. (2010). A comparison of Latin hypercube and grid ensemble designs for the multivariate emulation of an Earth system model. *Computers & Geosciences*, 36(6):746–755.

Valpola, H. (2015). From neural PCA to deep unsupervised learning. *In Advances in Independent Component Analysis and Learning Machines*, pages 143–171.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P. A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML-08)*, pages 1096–1103.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P. A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408.

Aalto-DD 209/2018

ISBN 978-952-60-8258-5 (printed)
ISBN 978-952-60-8259-2 (pdf)
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Aalto University
School of Science
Department of Computer Science
www.aalto.fi

BUSINESS +
ECONOMY

ART +
DESIGN +
ARCHITECTURE

SCIENCE +
TECHNOLOGY

CROSSOVER

DOCTORAL
DISSERTATIONS