

Bayesian Variational Methods in Carbon Storage Monitoring



Kristian Gundersen

Thesis for the degree of Philosophiae Doctor (PhD)
University of Bergen, Norway
2021

UNIVERSITY OF BERGEN



Bayesian Variational Methods in Carbon Storage Monitoring

Kristian Gundersen



Thesis for the degree of Philosophiae Doctor (PhD)
at the University of Bergen

Date of defense: 19.01.2021

© Copyright Kristian Gundersen

The material in this publication is covered by the provisions of the Copyright Act.

Year: 2021

Title: Bayesian Variational Methods in Carbon Storage Monitoring

Name: Kristian Gundersen

Print: Skipnes Kommunikasjon / University of Bergen

Preface

This study was carried out at the Department of Mathematics, University of Bergen. This work is part of the project ACTOM, funded through the ACT programme (Accelerating CCS Technologies, Horizon2020 Project No 294766). Financial contributions made from: The Research Council of Norway, (RCN), Norway, Netherlands Enterprise Agency (RVO), Netherlands, Department for Business, Energy & Industrial Strategy (BEIS) together with extra funding from NERC and EPSRC research councils, United Kingdom, US Department of Energy (US DOE), USA. I have been supported by the Research Council of Norway, through the CLIMIT program (project 254711, BayMode) and the European Union Horizon 2020 research and innovation program under grant agreement 654462, STEMM-CCS. I also would like to acknowledge NVIDIA Corporation for providing their GPUs in the academic GPU Grant Program.

Advisory committee

- Guttorm Alendal (University of Bergen, Department of Mathematics)
- Jarle Berntsen (University of Bergen, Department of Mathematics)
- Hans Julius Skaug (University of Bergen, Department of Mathematics)

Outline

This thesis consists of an introductory part and three scientific papers. Chapter 1 contains an introduction to monitoring of CCS sites and describes challenges associated to monitoring. Chapter 2 addresses basic concepts of deep learning. In Chapter 3 we introduce Bayesian Neural Networks and Variational Inference as tools for assessing uncertainty. The last chapter of the introductory part is Chapter 4 and presents how dropout in a neural network can approximate Variational Inference and different variational auto-encoder models. A brief summary of the papers is given in Chapter 5.

List of Papers

Paper A:

Kristian Gundersen, Guttorm Alendal, Anna Oleynik, Nello Blaser, (2020) *Binary Time Series Classification with Bayesian Convolutional Neural Networks when Monitoring for Marine Gas Discharges*, Algorithms **13/6**

Paper B:

Kristian Gundersen, Anna Oleynik, Nello Blaser, Guttorm Alendal (2020) *Semi Conditional Variational Auto-Encoder for Flow Reconstruction and Uncertainty Quantification from Limited Observations*, arXiv preprint arXiv:2007.09644 (Submitted to Physics of Fluids august 2020)

Paper C:

Kristian Gundersen, Seyyed Hosseini, Anna Oleynik, Guttorm Alendal (2020) *A Variational Auto-encoder for Reservoir Monitoring*, arXiv preprint arXiv:2009.11693 (To be submitted to Machine Learning)

Acknowledgements

After three and a half years with fun, frustration, stress and ups -and downs there are many I want to thank as it nears the end of this project. First of all, I want to thank my supervisors Guttorm Alendal, Jarle Berntsen and Hans J. Skaug. Especially I want to thank my main supervisor Guttorm. Even though that I have shown up outside your office at all hours with crazy ideas and suggestions and no filter, I have always felt welcome. I have highly appreciated your patience, guidance, interesting discussions, comments and advice.

Big thanks go(s) to my friend and colleague Anna Oleynik. We have been working closely together on the STEMM-CCS project, and I especially remember with joy our trip to Melbourne, Australia for the GHGT-14 conference. It was great fun and I have really appreciated your company, advice, eye for details, sarcastic and non-sarcastic comments and discussions the last few years. I also have to direct a great thank to "The Machine Learning Guru", aka Nello Blaser, for valuable input, discussions and comments to my papers. Thanks to Anna for reading through my thesis, and Nello for valuable suggestions.

I have been so privileged that I have had the opportunity to travel and bring my family with me on two research visits. The first to Plymouth, England and the second to University of Texas Austin, US. I want to thank for the warm welcome and hospitality at both places. It was educational and a fun experience. Thanks to Seyyed Hoessini for the cooperation, comments and hospitality during my research visit to Bureau of Economic Geology.

Making these trips and travels has been made possible through the support from the projects I have been a part of: STEMM-CCS, BayMoDe and ACTOM. Through these projects I have get to know an interesting community and it has given me many opportunities to learn new things. I want to direct my gratitude towards the projects and everybody involved.

Furthermore, I want to thank the entire Department of Mathematics, it has been a good place to work. I thank both Sondre and Håvard for unintentionally addressing my issue of slight obesity, by bringing me to hiking trips and encouraging me to take the stairs, but also for numerous coffee and lunch breaks. A great thank goes to my nearest neighbor at the office through these years Shirin, for fun and interesting discussions and teaching me things like what "ab" means on Persian.

I have been fortunate to be surrounded with great friends and family. Thanks to all of you making the time of my study memorable. Thanks

Special thanks go to my parents Mette and Bjarne. You have always believed in me, supported and encouraged me throughout my PhD studies and in life in general. I also want to direct my gratitude towards my parents in law (to be) Grethe and Sigvald, who have always been helpful and supportive.

Finally, the one who deserves the greatest thanks of all is my fiancé and the love of my life Solveig. You have taken care of our beloved kids Aryan and Leah when I have been struggling with notations and bugs in my code. You are the most caring, kind and generous person I know, and you have supported me, encouraged me, helped me and loved me, so that this work has been possible. I am extremely grateful!

Kristian Gundersen
Bergen, 30.09.2020

Contents

Preface	i
Outline	ii
List of Papers	iii
Acknowledgements	iv
I Background	1
1 Monitoring of Carbon Capture and Storage (CCS)	3
1.1 Introduction to CCS	3
1.2 Marine CCS Monitoring	4
1.3 Subsurface CCS Monitoring	8
2 Deep Learning	9
2.1 Artificial Neural Network	9
2.2 Objective functions and optimization	10
2.3 Validation of ANNs	12
2.4 Convolutional Neural Networks(CNN)	13
2.5 Autoencoders	14
3 Uncertainty quantification in Deep Learning	17
3.1 Bayesian Neural Networks (BNN)	18
3.2 Monte Carlo (MC) Estimators	19
3.3 Variational Inference (VI)	20
3.3.1 Reparametrization of the VI Objective	21
4 Variational Methods in Deep Learning	23
4.1 Dropout Neural Networks	23
4.1.1 Dropout Neural Network approximates VI	25
4.2 Variational Autoencoders (VAE)	26
4.3 Conditional Variational Autoencoders (CVAE)	28
4.4 Semi Conditional Variational Autoencoders (SCVAE)	29
4.4.1 SCVAE on MNIST Dataset	30
5 Introduction to the papers	35

6 Contribution and Outlook	37
Bibliography	44
II Included Papers	45
Paper A:	
Binary Time Series Classification with Bayesian Convolutional Neural Networks When Monitoring for Marine Gas Discharges	47
Paper B:	
Semi Conditional Variational Auto-Encoder for Flow Reconstruction and Un- certainty Quantification from Limited Observations	73
Paper B:	
A Variational Auto-encoder for Reservoir Monitoring	109

Part I
Background

Chapter 1

Monitoring of Carbon Capture and Storage (CCS)

The purpose of this section is to briefly introduce CCS and discuss the importance and difficulties of designing monitoring programs for both a marine and a subsurface environment. The focus is on providing a deeper insight into CCS to better enable understanding of the papers presented in Part II. The methods and models developed in this thesis are general and can be used in various contexts. However, the applications we base our methods on are related to CCS monitoring. Therefore, it is appropriate to introduce the CCS technology and important aspects related to monitoring.

1.1 Introduction to CCS

CCS is a technology that captures CO₂ at the source to inhibit increased levels of CO₂ in the atmosphere, with the ultimate purpose of mitigating climate change. Important industries such as cement production are huge point source emitters of CO₂, and there are no real alternatives other than CCS for reducing the release of CO₂ into the atmosphere. According to International Energy Agency and The Intergovernmental Panel on Climate Change be a key factor in reaching the sub-1.5°C goal and should consist of 14% of the total CO₂ reduction [2, 21]. Utilizing the huge offshore storage capacities is a necessity.

CCS involves three steps: capture, transport and storage of the CO₂. The capture phase can be simple or difficult depending on the emitters, however, in most cases it is necessary to mix flue gas with a liquid solvent that reacts, either physically or chemically with the CO₂. From the mixed solution it is possible to extract only the CO₂ by either adding heat (chemical absorption) or by lowering the pressure (Physically absorption). After separation, the CO₂ is purified and compressed, and then transported to a storage site for permanent storage. The transport is preferably by pipelines, however, transport with large cargo/LNG ships, trains or trailers are feasible alternatives. In fact, the *Langskip* project [53] in Norway intends to capture CO₂ from different emitters on the East coast of Norway and transport it by large ships to the west coast of Norway, where suitable storage formations are available. Chemical absorption involves adding liquid solvent that reacts with the CO₂, thus capturing the CO₂ in this new output. The

CO₂ can be extracted from the new solution, for instance, by increasing the temperature of the resulting fluid. Physical absorption is the process of adding a fluid to the flue gas that separates the CO₂ without a chemical reaction. Other techniques such as absorption with solids, membranes or cryogenic separation are under development.

The only feasible alternatives of storing captured CO₂ on a large scale is in geological formation at least 800 metres below the surface of the earth. This depth requirement is important since the CO₂ under high pressure will remain in a liquid state and, as a result, will be less mobile than it would be in a gaseous state. Storage sites should preferably be high permeable reservoirs that are confined with a cap rock. The CO₂ will migrate upwards, and without a proper cap-rock to confine the CO₂ it will eventually leak into the atmosphere. This is the same principles that has confined oil and gas for millions of years. In time large portions of the CO₂ mineralize and the risk of substantial release from the reservoir will be significantly reduced. A schematic overview of these steps is presented in Figure 1.1. For a review of the CCS technology we refer the reader to e.g. [25, 56]

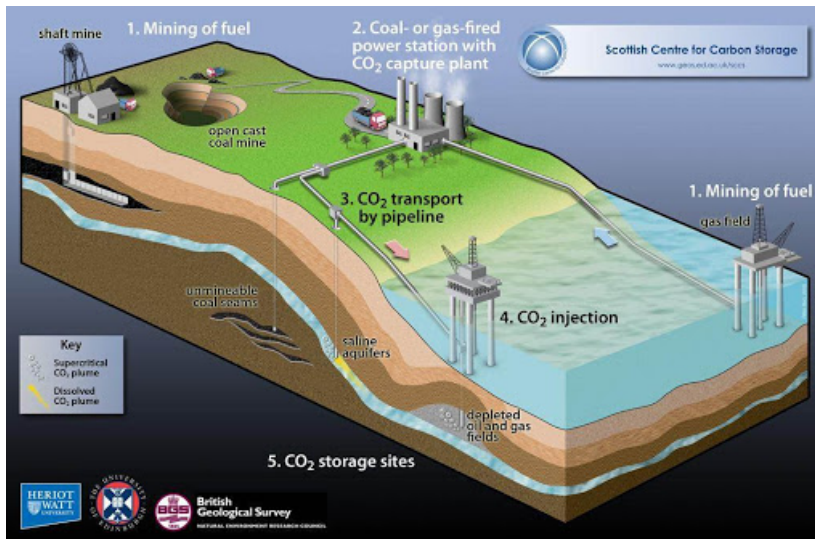


Figure 1.1: The figure illustrates the CCS process. Fuel is mined and delivered to a power plant that, in addition to energy, produces CO₂. The CO₂ is captured, and transported to an injection site. From the injection site, the CO₂ is injected into a reservoir, thus avoiding the CO₂ from the power plant to reach the atmosphere.

1.2 Marine CCS Monitoring

Monitoring is the procedure of observing and supervising/checking the progress of a process over a period of time and keeping the process under systematic review. Environmental monitoring involves collecting data from sensors, taking samples or remote sensing (e.g. satellite data), that can provide information about changes from the baseline. In general a proper marine monitoring program consists of three important aspects,

detection, quantification and localization.

Monitoring of the marine environment is a part of the ecosystem based Marine Spatial Planning [18] and Life under Water is one of UN's Sustainable Development Goals. Ensuring safe and secure offshore storage requires extensive marine monitoring programs over a long period of time. CCS projects will be designed to keep the stored CO₂ within the intended formations, and the injection wells and the formation will be monitored by standard technologies to ensure detection of unanticipated events [61]. However, due to the large amount of CO₂ that needs to be stored and, as a result, the large area that needs to be monitored, there is always a possibility that CO₂ may migrate toward the sea floor undetected. As a precaution, the marine environment will have to be monitored for indications of a leak. Monitoring of the seabed is also necessary to comply with the regulations ¹.

The marine component of the monitoring program assures that a storage project can coexist with other offshore activities. The associated environmental monitoring can also be beneficial for other purposes. For instance, tools are under development for assessing the total environmental stress imposed on the oceans, e.g. Cumulative Effects Assessments (CEA), in view of Marine Spatial Planning and ecosystem services framework [35, 47, 63], and the potential stress added from CO₂ storage projects needs to be documented. The marine monitoring program also precludes unjustified allegations of having adverse environmental effects [12], but will impose additional costs and challenges to the storage project [6, 7, 54]

Environmental changes, e.g. changes in bottom fauna or in the pelagic ecosystem [8, 70], detection of bubbles from ship sonars [13, 52], or elevated concentration of dissolved gases [4, 10, 19, 68], can be used as indicators of marine gas releases [16]. However, the real challenge is the high variability of the marine environment, both in current conditions [3] and in biochemical activities [10]. Monitoring an unsteady marine environment for changes in variables that are naturally present can be considered as a classification problem: data streams need to be categorized as leak or non-leak. A false positive, i.e. indications of a leak that is not present, can become costly. The monitoring program will enter a mode to initiate actions to locate and confirm the leakage. This might include relatively cheap analyses of existing data or costly cruises and surveys for confirmation. Another aspect is undetected seepage, so called false negatives. A monitoring program has to be designed to minimize such incidents as they may impose undetected additional stress on the environment.

In this thesis we focus on two distinct topics related to marine monitoring, namely anomaly or leakage detection and impact assessment or quantification. If we are unaware that an abnormal event has occurred, it is impossible to initiate counter measures. Furthermore, it is important to assess how the leak will impact the environment, i.e. how the CO₂ spreads through the water column to ascertain what measures to implement.

¹<https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32009L0031&from=EN>

Anomaly detection

To confirm a CCS leakage in a marine environment we must first have a system that detects, and clearly signals that the leak is present. Blowouts will be easily detected due to the natural extent and severity of the incident. However, if the CO₂ seeps out through cracks and faults, the task becomes much more difficult. In this context, machine learning algorithms may increase the ability to detect a leakage.

A sub-field of machine learning called anomaly detection is devoted to detecting rare events. To distinguish data that stands out from the bulk of the data as whole is referred to as anomaly detection or outlier detection. There are different algorithms developed for anomaly detection and that are applied to fraud detection in insurance, detecting ecosystem disturbances, detection of deceases in medical applications and applications related to network intrusion and network attacks.

In general there are three main approaches for detecting anomalies in data, unsupervised, supervised and semi-supervised anomaly detection. In supervised anomaly detection, we have knowledge of whether or not this particular instance is abnormal or normal. Given this information, it is possible to train a model to classify new data without labels as either abnormal or normal. This is a typical binary classification problem. Unsupervised anomaly detection uses unlabelled data to identify anomalies. Here instances that fit least can be viewed as anomalies. Training a model, and subsequently testing whether a new instance arises from this model is called semi-supervised anomaly detection.

Marine monitoring with geochemical sensors can be conducted with either under water vehicles or by fixed installations on the seafloor. In this thesis the focus is on fixed installations. Fixed sensors can produce data of the CO₂ concentration at specific, but sparse locations over time. While we can obtain the data that corresponds to naturally variable CO₂ concentration, a leakage from a CCS confinement has never occurred. Therefore, models are needed to create a dataset for credible leakage scenarios that can be used to represent the abnormal situation, i.e. where a leakage has occurred. This is in fact what we investigated in Paper A, where we use a probabilistic deep learning model to detect CCS leakages.

Quantification and localization

To initiate appropriate action after a leakage is confirmed, it is of crucial importance to quantify and localize the leakage. The transport of CO₂ in the water column is governed by two important principles, advection and diffusion. The main driver of the transport is advection, i.e. mechanical transportation due to the current's conditions. Under the assumption that the excess CO₂ does not alter the density of the water, it is possible to model the CO₂ transport as a passive tracer. The advection-diffusion model for a passive tracer transport can be given as

$$\frac{\partial c(x,t)}{\partial t} = D\Delta c(x,t) - W(x,t) \cdot \nabla c + f(x,t), \quad x \in \Omega, \quad t \in [t_0, t_0 + T] \quad (1.1)$$

with some appropriate boundary and initial conditions. Here Ω is a bounded connected domain in \mathbb{R}^d , $d = 2, 3$, $c(x, t)$ is the concentration of CO_2 , $D(x, t) \geq 0$ is the diffusion coefficient, $W(x, t) \in \mathbb{R}^d$ is the velocity field and $f(x, t)$ is the source term. Oleynik et. al [55] used advection-diffusion model to optimize sensor layout in a CCS monitoring setting.

The advection diffusion model requires velocity fields often obtained by running ocean models such as FVCOM [14] or Bergen Ocean Model (BOM) [5]. With the advection-diffusion model it is possible to estimate the transport of CO_2 in the water column; however, this tool requires that velocity fields be defined over the entire monitoring domain. With only sparse measurements available, one of the key challenges is to go from the observations to a representation across the entire monitoring domain. Due to the probabilistic nature of the forcing of the ocean models (e.g. weather conditions), the generated velocity fields are factors of uncertainty. To reflect this uncertainty we want to obtain probabilistic velocity fields. It is possible to simulate currents with different forcing to obtain different velocity field under different conditions. This is a typical Monte Carlo estimator (see Section 3.2) for modelling the velocity field uncertainty. The problem is that these simulations are extremely costly, and the possible configurations of the forcing are vast. A data assimilation technique such as Ensemble Kalman Filters (EnKF) [20] combines models, observations and Monte Carlo simulations to obtain probabilistic representation of the entire domain. However, they suffers from the above-mentioned issues.

Data driven methods is another approach to create velocity field given sparse observations. From existing data, it is possible to estimate parameters of a statistical model that outputs velocity fields given the measurements. A conventional approach for this purpose is, e.g. the Gappy Proper Orthogonal Decomposition (GPOD) method. The challenge for the GPOD-method is that it does not scale well towards large data sets. Traditional auto-encoders have been used for this purpose [1]; however neither the GPOD nor the Auto-encoder generates probabilistic velocity fields. We have developed a method for probabilistic reconstruction dependent only on the measurements. After the model is trained, it is possible to input new unseen measurements to generate probabilistic velocity fields for the entire monitoring domain. Although optimization of the parameters of the model is time consuming, the prediction is not.

With the computationally low-cost advection-diffusion and probabilistic reconstruction model we have a framework that can be valuable in quantification and localization of leakages. The absence of directly dependence on the governing forcing makes this framework suitable for fast determination of the severity and extent of an incident. This will be of crucial importance to limit the negative impact from a possible CCS leakage and to provide input for proper support for decision makers.

1.3 Subsurface CCS Monitoring

We also address monitoring of CCS site from a subsurface perspective. The Above Zone Monitoring Interval (AZMI) [46] is the area above the cap rock of a CCS injection reservoir, see Fig. 1.2.

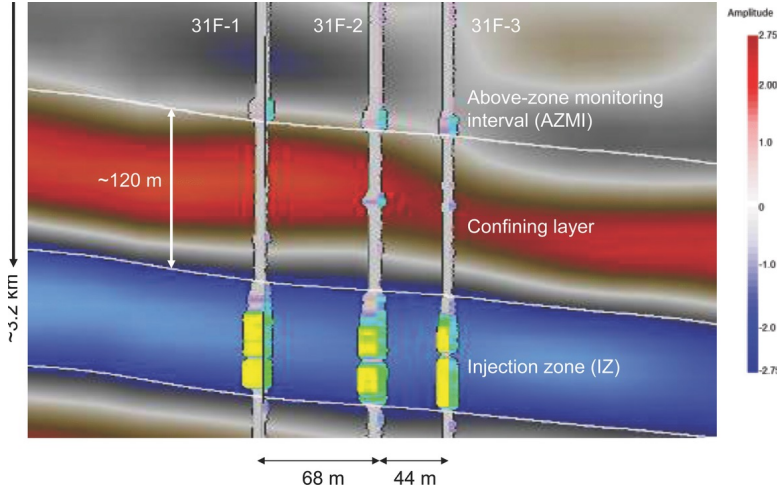


Figure 1.2: Illustration of the AZMI, the confining and injection layer, at a CCS injection site with one injection well 31F-1, and two observation wells 31F-2 and 31F-3. Figure is taken from [38].

Monitoring in the above zone is a method by which properties in the AZMI are recorded with a limited number of AZMI-wells to look for changes as an indicator of potential migration of CO_2 from the storage formation. CO_2 is buoyant and will rise towards the seafloor, and possibly affect the AZMI. In case of a leak from the reservoir, for example, we anticipate the pressure will increase. By analysing the pressure, it is possible to both detect and quantify leakages from the CCS reservoir with measurements in the AZMI.

The pressure in the AZMI is stable, even with injection of CO_2 . In contrast to marine monitoring, the natural variability in the AZMI is low. Hence, detecting anomalies is easy. To quantify and pinpoint where a leakage however, is a much more difficult task, and presents the same challenges as in the case of marine monitoring.

It is possible to simulate leaks in the injection zone and how it affect the pressure in AZMI. As for the marine monitoring case, observations or AZMI-wells are sparse. If we were able to use these measurements, and reconstruct the pressure field in the entire reservoir, it is possible to pinpoint the location of the leak by identifying where the pressure is the greatest. Simulations of different leakage scenarios, with different fluxes can be input for a deep learning model that recreates the pressure and classifies the flux of the leakage. In-situ data can then be given as input during prediction, classifying the flux and recreating the pressure field.

Chapter 2

Deep Learning

The models we have developed for monitoring purposes rely on deep neural network models. Hence, we introduce the core concepts of deep learning. This includes, but is not limited to, the definition of a 2-layer feed forward neural network, cost and objective functions, regularization and the manner in which a network can be trained to produce desired output. The main references for this chapter are the textbook "Deep Learning" by Goodfellow et. al [29] and the work of Gal [22].

2.1 Artificial Neural Network

Let $\mathbf{x}^{(i)}$ be an instance of data which is input to a deep learning model, with associated target values $\mathbf{y}^{(i)}$. A target value can be a class or as we will use later the data $\mathbf{x}^{(i)}$ itself. All the instances of $\mathbf{x}^{(i)}, i = 1, \dots, N$, constitute the data set \mathbf{X} , and all the target values $\mathbf{y}^{(i)}, i = 1, \dots, N$, comprise the data set \mathbf{Y} .

A feed forward neural network is a hierarchical model that consists of nodes or computational units divided into subsequent layers. For each node, a non-linear activation function is applied. The nodes between each layer are connected, so that the input to a node is totally dependent on the output from the nodes of the previous layer. The model is called a *deep learning model* if there are multiple hidden layers; see Section 2.1. The simplest deep learning model has at least one hidden layer: an input layer and an output layer. This structure makes it possible to formulate the deep learning model as a linear system of equations.

The model input to a neural network is here defined as vector $\mathbf{x}^{(i)}$ with Q elements. The input is transformed linearly by \mathbf{W}_1 and \mathbf{b} such that $\mathbf{f}(\mathbf{x}^{(i)}) = \mathbf{W}_1 \mathbf{x}^{(i)} + \mathbf{b}$. \mathbf{W}_1 transforms the input to a vector of K elements and is often called the weight matrix, while the translation \mathbf{b} is referred to as the bias. The bias can be interpreted as a threshold for when the neuron activates.

A nonlinear activation function is applied elementwise to the transformed data. The activation function is typically given as a rectified linear unit (ReLU) [48] or tanh function. This activation introduces non-linearity to the other linear operations. The superposition of the linear and nonlinear transformation is in combination with the ac-

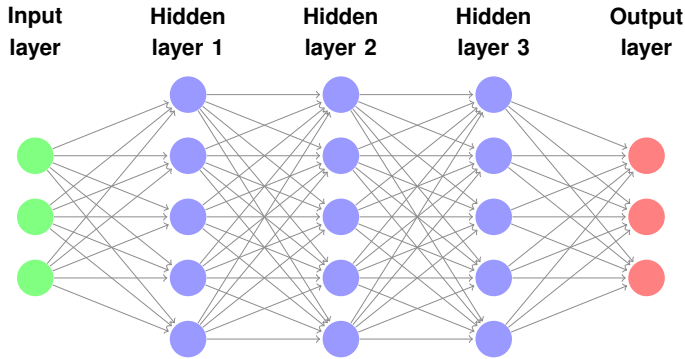


Figure 2.1: Illustration of a deep neural network with three hidden layers.

tivation function and is what we refer to as a hidden layer.

Applying another linear transformation \mathbf{W}_2 to the hidden layer results in this case to the model output or output layer. The size of the output layer is a row vector with D elements. Generally, many transformations and activations can be applied consecutively which will result in a more complex hierarchical model. A generalization to a network with several hidden layers is straightforward; to make this clear we here limit the notation to a single hidden layer. We note that $\mathbf{x}^{(i)}$ is vector of size Q , \mathbf{W}_1 is a $K \times Q$ matrix that transform s the input to K elements, \mathbf{W}_2 is a $D \times K$ matrix, transforming the vector into D elements and \mathbf{b} consists of K elements. We write this as linear system of equations transformed with the activation function σ

$$\hat{\mathbf{y}}^{(i)} = \mathbf{W}_2(\sigma(\mathbf{W}_1\mathbf{x}^{(i)} + \mathbf{b})) := \mathbf{f}^\omega, \quad \omega = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}\} \quad (2.1)$$

Depending on how the output layer is defined, we can use the network for classification or regression tasks. For classification purposes, the number of nodes in the output layer equals the number of classes, and typically transformed with a softmax function [26]. The softmax function is a generalization of the logistic map that normalizes the output relative to the different classes. In regression problems we want to estimate relations between variables; we want to predict a continuous output based on some input (variables). To use a linear activation function on the output layer will serve this purpose.

It has been shown that ANN is a universal approximation [33]; thus, our goal is to find the weights of the given network to best approximate the map from the input to the output. This means that we want to estimate the weights of the ANN ω , given the input data $\mathbf{x}^{(i)}$, the target $\mathbf{y}^{(i)}$ such that the predictions $\hat{\mathbf{y}}^{(i)}$ is minimized towards the true target values $\mathbf{y}^{(i)}$. This is a typical optimization problem, which can be minimized with an objective function and optimization procedure.

2.2 Objective functions and optimization

An objective function for use in deep learning typically contains two terms: cost function and regularization. The cost function takes the predicted and the true values as input.

Depending on the task and what one wants to minimize, the cost function maximizes a likelihood. In classification problems this is can be the negative cross entropy

$$\mathcal{C}_1^{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}}(\mathbf{X}, \mathbf{Y}) = -\frac{1}{N} \sum_{j=1}^N \mathbf{y}_j^{(i)} \log(\widehat{\mathbf{y}}_j^{(i)}) = -\log p(\mathbf{Y}|\mathbf{f}^\omega(\mathbf{X})), \quad (2.2)$$

and in regression problems the Mean Squared Error (MSE)

$$\mathcal{C}_2^{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}}(\mathbf{X}, \mathbf{Y}) = \frac{1}{N} \sum_{i=1}^N (\mathbf{y}^{(i)} - \widehat{\mathbf{y}}^{(i)})^2 = -\log p(\mathbf{Y}|\mathbf{f}^\omega(\mathbf{X})), \quad (2.3)$$

Minimization of the negative cross entropy and the MSE is well known to be equivalent to minimize the negative log likelihood of the parameter estimation [66] for neural networks. Depending on the task, minimizing Eq. (2.2) or Eq. (2.3) with respect to the parameters $\omega = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}\}$ maximizes the likelihood of these parameters. The choice of the cost function is not restricted to those given above, and depend on the data, the model structure and what one wants to predict with the model.

One of the key problems in deep learning is a phenomenon called overfitting. Overfitting occurs if the optimized model performs poorly on new unseen data, i.e. it does not generalize well. To address this problem, regularization is added to the cost function.

Regularization is a general technique, where the goal is to make an ill posed problem well-posed. Overfitting is basically one example of an ill-posed problem. For optimization problems, you could add a penalizing functional: L2 or L1 norm for the parameters; or use dropout.

Regularization in ANNs work by penalizing the cost function, e.g. forcing the weights to become small. The idea behind a specific regularization term could be to minimize the weights of the ANN to generate a simpler model that helps against overfitting. L2 regularization multiplied with some penalizing factor λ_i is one of the most common regularization techniques. The cost function with regularization is called the objective function. Adding L2 regularization to equation Eq. (2.2) or Eq. (2.3) result in the objective function

$$\mathcal{L}(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) = \mathcal{C}^{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}}(\mathbf{X}, \mathbf{Y}) + \lambda_1 \|\mathbf{W}_1\|^2 + \lambda_2 \|\mathbf{W}_2\|^2 + \lambda_3 \|\mathbf{b}\|^2 \quad (2.4)$$

Another way of regularizing the cost function is through dropout, which is a stochastic regularization technique. In Section 4.1 we will elaborate on dropout as a regularization method and how we can use it to quantify uncertainty in predictions.

Minimizing the objective in (2.4) with respect to the weights ω with an objective function and a gradient descent optimization method has proven to give good results in a wide range of applications.

The gradient descent method [15] updates the parameter ω using the entire data set

$$\omega_t = \omega_{t-1} - \eta \nabla \mathcal{L}(\omega_{t-1}). \quad (2.5)$$

Here ω_t represents the current configuration of the weights, while ω_{t-1} represents the previous one. The parameter η is referred to as the learning rate, i.e. how large the step in the negative gradient direction the update of the weights should be. Too small steps can lead to poor convergence, while too large steps can lead to overshooting, i.e. missing local/global minimums. Usually it is too expensive to calculate the gradient over the entire dataset. This is solved by a technique called stochastic gradient descent [57]. Stochastic gradient descent performs a parameter update for each training example. A natural extension and a more cost-efficient approach is the mini-batch gradient descent approach. In mini-batch optimization, the gradient is approximated by calculating the mean of the gradients on sub-sets or batches of the entire data set,

$$\omega_t = \omega_{t-1} - \frac{\eta}{n} \sum_{i=1}^n \nabla \mathcal{L}_i(\omega_{t-1}). \quad (2.6)$$

The mini-batch gradient descent iterative process can be implemented in the neural network with the back-propagation algorithm [58]. In back-propagation, the weights are updated through a forward and backward pass. In the forward pass, we predict with the current weight configuration and compare towards the target values. In the backward pass, we use the chain rule successively from the output to the input to calculate the gradient of ω . Based on the gradient direction and the learning rate, the configuration of the weights is updated.

One of the disadvantages of the vanilla gradient descent approach to the ANN optimization problem, is that it has a fixed learning rate η . In line with the development of ANN, methods dedicated to deep learning and adaptive adjustment of the learning rate have been developed. Besides SGD with momentum [62], the two most used optimization methods for ANNs are ADAM [39] and Root Mean Square Propagation (RMSProp) [65]. RMSProp adaptively adjusts the learning rate of the gradients based on a running average for each of the individual parameters. The ADAM-algorithm individually adjusts the weights in terms of both the running average, but also with respect to the running variance.

The use of back-propagation together with a stochastic gradient descent method, increase in available data and hardware have been the successes of deep learning during the past decade.

2.3 Validation of ANNs

To validate and ensure that the predictions of the deep learning model also performs well on new unseen instances, the data is split into three independent sub-data sets: a training, a validation and a test data set. The training data set is directly used to optimize the parameters of the model. The validation data set is indirectly used to optimize the model, that is, we monitor the performance on the validation dataset after each epoch. An epoch is one pass in the optimization over the entire training dataset. During training, the model sees the same training data multiple times, however, the instances are usually randomly shuffled before a new epoch starts.

After each epoch, we predict with this temporally model on the validation data set. Usually we put criteria on the performance on the validation data set for when to stop the optimization. We can use a so called early stopping regime, where the model stops training if it does not see improvement on the validation score after a certain number of epochs without improvement.

The purpose of the test data set is to validate on new unseen data that has not been part of the training or the continuous validation of the model.

Outputs from simulations or monitoring sensors, i.e. time series data can be used as input to a deep learning model. The temporal domain can be used to obtain instances. For splitting of training, validation and test data sets in time dependent processes, we distinguish between two splitting strategies. The data can either be split randomly or dependent in time. Splitting randomly in time will lead to instances that are similar in both the train, validation and test data, due to the autoregressive properties of the data. This might lead to overfitting of the deep learning model. Splitting the data time dependent procedure, means that the data is divided into subsets where the data in the each data set is a closed interval with respect to the temporal domain. E.g. we can use the first 80% of the data as train, the next 10% as validation, and the last 10% as test data set. An issue with this approach is that if the process governing the data is complex, e.g. shifts between different stochastic processes, this splitting might lead to a model that is out of test distribution. That is, the test and training data is to different from each to generate meaningful predictions on the test data set. One the other hand, with increasing amount of data, the out of test data problem could be resolved.

In all papers we have utilized convolutional neural networks. The next section describes the principle of CNNs and the mechanism behind their success.

2.4 Convolutional Neural Networks(CNN)

CNNs [44] utilize the grid structures in the data to be analysed, i.e. in 1-D the regular sampling in time series data, or in 2-D the fixed structure of pixels in an image, and use convolutions instead of matrix multiplication in at least one of its layers. The discrete convolution operation in 2D is defined as

$$s(m,n) = (x * h)(m,n) = \sum_{i,j} x(i,j) \cdot h(m-i,n-j) \quad (2.7)$$

where x is a $M \times N$ matrix, $h(m,n)$ is a $K \times L$ matrix, and i,j range over all legal subscripts. The resulting convolution s is a $(M + K - 1) \times (N + L - 1)$ matrix. The observation x is averaged with the kernel $h(k,n)$, to produce a generally less noisy output $s(m,n)$. The output of a convolution operation is often referred to as features.

Normally implementations of CNNs do not actually use regular convolution, but cross correlation operation. Cross-correlation and convolutions are very alike, and the major difference in definition is that the kernel $h(m-i,n-j)$ is altered to $h(m+i,n+j)$. During a convolution operation we have to reverse the order of either the input or the

kernel, while this is not necessary in the cross-correlation operation. Using cross-correlation simply reduces the computational cost during optimization of the CNN.

There are three main mechanisms for why convolutions and CNNs are successful; sparse interactions between nodes, parameter sharing and equivariant representation [27, 44].

- 1) In feed forward neural networks (matrix multiplication neural networks), every output unit interacts with every input. CNNs is different in terms of often having sparse interactions. Sparse interaction means that the output is dependent on a limited number of inputs. However, with multiple consecutive CNN layers, the units become indirectly connected. Due to the indirectly connected units or sparse interactions the CNNs can efficiently describe complicated interactions between several variables. Sparse interaction is a result of the convolutional operation and occur when $|h| < |x|$ [27].
- 2) If a parameter is used by several functions in a neural network model, this is referred to as parameter sharing. Instead of learning a separate set of parameters for every location, CNNs learn one set that can be utilized everywhere. Sharing parameters limits the architecture of the model, reduces memory requirements and improves the quality of the model/estimator.
- 3) The convolutional operation is equivariant to translation [27]. Equivariant means that if the input changes, the output changes in the same way. Translation means shifting the input. Equivariant to translation simply means that shifting the input will not alter the output. The CNN generates a record of different features and the features will be represented similarly regardless of where they appear.

The architecture of a full CNN consists in general of three important steps that are repeated: the convolution operation, a non-linear transformation via the activation function and a pooling operation. Pooling is a down-sampling technique to extract important features from the convolutional operations. The step size which the kernel slides over the input is called stride. If the stride is larger than one, the dimension of the output decreases. Hence, using strides with a step size of two or more is often used instead of pooling operation.

2.5 Autoencoders

A vanilla autoencoder is a neural network that basically copies the input to the output. The autoencoder consists of an encoder $h = f(\mathbf{x}^{(i)})$ and a reconstructor or decoder $g = h(\mathbf{x}^{(i)})$. A successful autoencoder can recreate the input such that $g(f(\mathbf{x}^{(i)})) = \mathbf{x}^{(i)}$. To be able to copy and recreate the exact same input is not particularly useful, and usually the architecture of autoencoders is built so that it is not able to map the input to the output perfectly. Let X be the input space, and Z be a range space of f , which is commonly referred to as a feature space. The encoder f maps the input to the feature

space, while the decoder g maps the features space back to X . i.e.,

$$\begin{aligned}f &: X \rightarrow Z \\g &: Z \rightarrow X.\end{aligned}$$

Often the feature space Z has a lower dimension than X which forces the autoencoder to withdraw the most important characteristics in the data. These autoencoders, which are called undercomplete autoencoders, have been used for dimensionality reduction and feature extraction. In fact, the decoder of an undercomplete autoencoder with linear activation functions is equivalent to PCA. The autoencoder with nonlinear activation functions has the ability to learn nonlinear relationships in the data and thus learn more valuable generalizations than the PCA [28].

If the dimension of X is equal to Z , we say that the autoencoder is overcomplete. Overcomplete autoencoders use regularization as a tool to learn important representations Z . (e.g. sparse autoencoder, denoising autoencoders, contractive autoencoders). The concept by which we seek to find a stochastic representation of \mathbf{z} such that we can represent the encoder as a distribution $p(\mathbf{z}|\mathbf{x}^{(i)})$, and conversely the decoder $p(\mathbf{x}^{(i)}|\mathbf{z})$ and use variational inference to approximate $p(\mathbf{z}|\mathbf{x}^{(i)})$, is called Variational Autoencoders (VAE) [40]. VAEs are discussed in detail in Section 4.2.

Chapter 3

Uncertainty quantification in Deep Learning

There are several methods for uncertainty quantification in deep learning. The delta method [67] is a classical method for uncertainty quantification in statistical models. It can be used to quantify uncertainty in deep learning by estimate the variance of a random variable through a Taylor expansion of the objective function. By calculating the inverse of the Hessian matrix it is possible to approximate the uncertainty of the parameters. The delta method has been applied in ANNs [37, 51], however, due to the high cost of calculating the Hessian for high dimensional parameter spaces, other methods have been preferred. Nilsen et. al. [51] recently proposed a new cost efficient procedure of approximating the Hessian in deep neural networks, which may increase the popularity of the delta method in ANNs in the future.

Another approach to obtain uncertainty estimates is so-called Deep Ensembles (DE) [43]. In DE many models are trained with different initialization of the weights. Together with the random nature of the mini-batch optimization, this leads to an ensemble of models with different configurations and predictive outcomes. The process resembles bagging or bootstrapping. By assessing the variance over model ensemble predictions, uncertainty estimates can be obtained. The simplicity of DE is the key advantage, while the disadvantage is the high computational cost.

Dropout neural networks [60] can be used for uncertainty quantification if dropout is kept on during prediction. This is referred to as MC-dropout [22]. In Paper A we used MC-dropout for uncertainty quantification in time series classification. It turns out that there is a close connection between MC dropout and BNN approximated with VI. MC-dropout and how it approximates VI is reviewed in detail in Section 4.1. In the remainder of this section we will review the concepts of Bayesian parameter estimation and Bayesian Neural Networks, and two important approximating methods for estimating the intractable integral that arise in a Bayesian framework.

3.1 Bayesian Neural Networks (BNN)

Bayesian approaches are methods for assessing uncertainty in a model output. In contrast with only obtaining point estimates, these approaches approximate a general distribution over the model parameters. With information about the uncertainty of the models parameters, it is possible to obtain uncertainty estimates of the models predictive performance. In general we distinguish between two types of contribution to uncertainty in Bayesian modelling: epistemic and aleatoric uncertainty [17]. Epistemic or model uncertainty is related to the how well a model can explain the data, i.e. uncertainty in the model parameters. Model uncertainty usually diminishes when data increase. Aleatoric or measurement uncertainty is related to the data, e.g. uncertainty introduced by noisy input or labels. BNNs allow for capturing the predictive uncertainty, that is not only the aleatoric uncertainty but also the epistemic uncertainty by estimating the posterior distributions. Here, we will briefly describe the concepts of Bayesian parameter estimation and Bayesian Neural Networks, and two approximating methods for estimating the intractable integral that often arises in a Bayesian framework.

Here we assume that optimal weights of the network are described by a probability description $p(\omega)$, that is needed to be estimated. We view the input data $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ and a target values data $\mathbf{Y} = \{\mathbf{y}^{(i)}\}_{i=1}^N$, as realizations of some random variables \mathbf{x} and \mathbf{y} . For simplicity we assume that \mathbf{X} and \mathbf{Y} contains i.i.d. samples.

Any knowledge we have on the weights beforehand are referred to as the prior and denoted $p(\omega)$. In neural networks we usually have no prior information about weights and $p(\omega)$ is often chosen to be Gaussian or Laplacian distribution. It can be shown that these priors result in different regularization effects. In particular, if we assume that both the BNN model error and the prior have Gaussian distributions, it can be shown that the Gaussian priors have a regularizing effect similar to L2-regularization. If the prior is Laplacian distributed, it can be shown that this is equivalent to L1-regularization. We will not go in details on the priors here, but we emphasizes the importance of the prior in BNNs and how it can potentially affect the estimation of the model parameters.

The prior is updated trough Bayes rule after observing the data (\mathbf{X}, \mathbf{Y}) :

$$p(\omega|\mathbf{X}, \mathbf{Y}) = \frac{p(\omega)p(\mathbf{Y}|\mathbf{X}, \omega)}{p(\mathbf{Y}|\mathbf{X})} \quad (3.1)$$

$p(\omega|\mathbf{X}, \mathbf{Y})$ is called the posterior distribution of ω , while $p(\mathbf{Y}|\mathbf{X}, \omega)$ is referred to as the model (here the neural network architecture) or likelihood function. Under the assumption that the instances in (\mathbf{X}, \mathbf{Y}) are independent, we can estimate the likelihood as the product of probabilities

$$p(\mathbf{Y}|\mathbf{X}, \omega) = \prod_{n=1}^N p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}, \omega) \quad (3.2)$$

The denominator $p(\mathbf{Y}|\mathbf{X})$ is referred to as the model evidence and is the marginal

likelihood with respect to the parameters ω ;

$$p(\mathbf{Y}|\mathbf{X}) = \int p(\mathbf{Y}|\mathbf{X}, \omega)p(\omega)d\omega. \quad (3.3)$$

In BNNs it is the model evidence that is usually intractable and needs to be approximated. The posterior predictive distribution is defined as

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \omega)p(\omega|\mathbf{X}, \mathbf{Y})d\omega, \quad (3.4)$$

where \mathbf{x}^* represents a new observation with unknown target \mathbf{y}^* . By varying ω , equation Eq. (3.4) can be viewed as an ensemble of models generated from $p(\omega|\mathbf{X}, \mathbf{Y})$. It is difficult and sometimes impossible to solve equation 3.4 analytically; thus we often resort to Monte Carlo sampling (See Section 3.2). If the Monte Carlo sampling becomes too computationally costly, we can turn to other approximation methods. These includes, but not limited to Laplace approximation [49], Stochastic expectation propagation [45] and Langevin diffusion methods [71].

In this thesis we use Variational Inference [36, 69] to approximate the intractable integrals that arise in BNNs. In the next sections we will briefly review both the standard Monte Carlo method and the basics of variational inference.

3.2 Monte Carlo (MC) Estimators

The most common way to estimate the posterior and the marginal likelihoods is with MC Estimators. The MC method [41] is a technique that uses random sampling to approximate the intractable posterior or integrals. The posterior can be considered as the expectation under the probability distribution $p(\omega|\mathbf{X}, \mathbf{Y})$

$$\mathbb{E}[p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y})] = \int p(\mathbf{y}^*|\mathbf{x}^*, \omega)p(\omega|\mathbf{X}, \mathbf{Y})d\omega. \quad (3.5)$$

Thus we can approximate $p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y})$ as

$$\mathbb{E}[p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y})] \approx \frac{1}{J} \sum_{j=1}^J p(\mathbf{y}^*|\mathbf{x}^*, \omega_j), \quad \omega_j \sim p(\omega|\mathbf{X}, \mathbf{Y}). \quad (3.6)$$

where J is the number of samples in the estimator. Under the assumption that the samples are i.i.d. and that the second order moment of $p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y})$ is bounded, the MC-estimator converges towards the true expectation for large enough J . In fact, the MC estimator in Eq. (3.6) is an unbiased estimator. Uncertainty quantification is possible by evaluating the empirical variance of $p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y})$.

A major drawback with the MC-estimator presented above is that if the sampling involves many parameters, the method has poor convergent properties. Sampling methods to solve this exist, e.g. Metropolis Hasting, Gibbs sampling, Hamiltonian Monte Carlo (HMC) [50]. These methods might improve the convergence properties compared to a random sampling MC-estimator, but usually not sufficiently for optimization of ANNs. In such cases variational inference may be a viable alternative.

3.3 Variational Inference (VI)

If ω has high dimension, and the posterior probability distribution over the parameters is too complex, estimating $p(\omega|\mathbf{X}, \mathbf{Y})$ based on sampling methods may be impractical and difficult, or at best have poor convergence properties. Variational inference addresses this issue by approximating the complex posterior $p(\omega|\mathbf{X}, \mathbf{Y})$, with a much simpler distribution $q_\phi(\omega)$ with variational parameters ϕ . See [9] for a full review of the Variational Inference method. VI makes use of the Kullback-Leibler (KL) divergence or relative entropy. KL divergence can be viewed as a measure of similarity between two distributions, and is defined as [42]:

$$KL[q_\phi(\omega)||p(\omega|\mathbf{X}, \mathbf{Y})] = \int q_\phi(\omega) \log \frac{q_\phi(\omega)}{p(\omega|\mathbf{X}, \mathbf{Y})} d\omega \quad (3.7)$$

Minimizing the KL divergence with respect to the variational parameters will thus approximate the true posterior density. Let $q_\phi^*(\omega)$ be a local or global minimum of the KL divergence in equation Eq. (3.7). We want to minimize the KL divergence with respect to the variational parameters to obtain $q_{\phi^*}(\omega)$ where

$$\phi^* = \arg \min_{\phi} KL[q_\phi(\omega)||p(\omega|\mathbf{X}, \mathbf{Y})] \quad (3.8)$$

We can rewrite the KL divergence in terms of the prior $p(\omega)$, marginal $p(\mathbf{Y}|\mathbf{X})$ and likelihood function $p(\omega|\mathbf{X}, \mathbf{Y})$, by combining equation (3.1) and (3.7) and some rearranging. We can express the KL-divergence as

$$KL[q_\phi(\omega)||p(\omega|\mathbf{X}, \mathbf{Y})] = \int q_\phi(\omega) \log p(\mathbf{Y}|\mathbf{X}, \omega) d\omega - KL[q_\phi(\omega)||p(\omega)] + \log p(\mathbf{Y}|\mathbf{X}). \quad (3.9)$$

Moving $\log p(\mathbf{Y}|\mathbf{X})$ to the left-hand side and changing signs defines the ELBO (or VI objective function)

$$\mathcal{L}_{VI}(\phi) \leq \log p(\mathbf{Y}|\mathbf{X}) - KL[q_\phi(\omega)||p(\omega|\mathbf{X}, \mathbf{Y})] \quad (3.10)$$

where

$$\mathcal{L}_{VI}(\phi) = - \int q_\phi(\omega) \log p(\mathbf{Y}|\mathbf{X}, \omega) d\omega + KL[q_\phi(\omega)||p(\omega)] \quad (3.11)$$

Since $KL[q_\phi(\omega)||p(\omega|\mathbf{X}, \mathbf{Y})] \geq 0$, we have that $\mathcal{L}_{VI}(\phi)$ is a lower bound of $\log p(\mathbf{Y}|\mathbf{X})$. Later, we will derive the ELBO for the specific variational auto-encoder model, which in essence has the same derivation as for Eq. (3.11). We refer the reader to Section 4.2 for a more detailed derivation of the ELBO. Maximization of the ELBO with respect to the variational parameters defining $q_\phi(\omega)$, are in fact equivalent to maximization of the log-likelihood $\log p(\mathbf{Y}|\mathbf{X})$. By maximizing the VI objective we can find the best approximating density $q_{\phi^*}(\omega)$ by

$$\phi^* = \arg \max_{\phi} \mathcal{L}_{VI}(\phi) \quad (3.12)$$

Optimizing $\mathcal{L}_{VI}(\phi)$ and finding $q_{\phi^*}(\omega)$ is a compromise between fitting the observed data properly by maximizing the likelihood, and minimizing the "distance" of prior

distribution $p(\omega)$ vs the variational distribution $q_\phi(\omega)$. By substituting the true posterior with the variational distribution $q_\phi^*(\omega)$, we can obtain an approximation of the true posterior distribution

$$\begin{aligned} p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) &= \int p(\mathbf{y}^*|\mathbf{x}^*, \omega)p(\omega|\mathbf{X}, \mathbf{Y})d\omega \\ &\approx \int p(\mathbf{y}^*|\mathbf{x}^*, \omega)q_\phi^*(\omega)d\omega \\ &\approx q_\phi^*(\mathbf{y}^*|\mathbf{x}^*). \end{aligned} \quad (3.13)$$

In even small BNNs, the posterior is either difficult or intractable to calculate.

3.3.1 Reparametrization of the VI Objective

An iteration during training of an ANN with VI, consists of a forward and backward pass for updating of the model parameters. A sample is drawn from the variational posterior distribution during the forward pass to evaluate Eq. (3.11), i.e. a stochastic sampling step. In the backward pass we need to calculate the gradients of ϕ . However, since ϕ is stochastic sampled in the forward pass it is not possible to directly calculate the gradient with the chain rule and backpropagation. To solve this issue we can reparametrize Eq. (3.11) with the so-called reparametrization trick introduced by Kingma and Welling [40]. The reparametrization trick samples from a parameter-free distribution and maps it to a deterministic function, where a gradient can be defined.

We can rewrite the VI objective function Eq. (3.11) in terms of the model output $\mathbf{f}^\omega(\mathbf{x}^{(i)})$ of the neural network with respect to the input $\mathbf{x}^{(i)}$

$$\mathcal{L}_{VI}(\phi) = - \sum_{i=1}^N \int q_\phi(\omega) \log p(\mathbf{y}^{(i)}|\mathbf{f}^\omega(\mathbf{x}^{(i)}))d\omega + KL[q_\phi(\omega)||p(\omega)] \quad (3.14)$$

A key issue optimizing the objective above is that if the dataset is large, we have to calculate the cost function for each instance, rapidly increasing the computational cost. In a gradient descent setting, this is solved with mini-batch optimization. An approximation to the cost function 3.14 can thus be expressed in terms of the sub-samples:

$$\widehat{\mathcal{L}}_{VI}(\phi) = -\frac{N}{M} \sum_{i \in S} \int q_\phi(\omega) \log p(\mathbf{y}^{(i)}|\mathbf{f}^\omega(\mathbf{x}^{(i)}))d\omega + KL[q_\phi(\omega)||p(\omega)], \quad (3.15)$$

with a random index set S of size M . The approximation above is an unbiased stochastic estimator of 3.14, i.e. $\mathbb{E}[\widehat{\mathcal{L}}_{VI}(\phi)] = \mathcal{L}_{VI}(\phi)$. Finding a local minimum of Eq. (3.14) is an approximation to the same local minimum of Eq. (3.15) [57]. This is a classical technique in deep learning optimization. The main challenge is that for Bayesian Neural Networks with more than one single hidden layer, calculation of the expected log likelihood in 3.14 is generally not tractable. This integral can be solved by Monte Carlo integration, and here we use Kingmas and Welling’s reparamerization trick [40] to deal with the problem.

The reparametrization trick introduces a new independent random variable ϵ used to reparametrize the weights. In each weight matrix $\mathbf{W}_{l,i}$ Kingma and Welling factorize the distribution of the rows $\mathbf{w}_{l,i}$. Reparametrization takes place by collecting $q_{\phi_{l,i}}^*(\mathbf{w}_{l,i})$ so that $\mathbf{w}_{l,i} = g(\phi_{l,i}, \epsilon_{l,i})$. The distribution over $\epsilon_{l,i}$ is often the standard normal distribution, however, any distribution $p(\epsilon_{l,i})$ can be specified. For short we write $p(\epsilon) = \prod_{l,i} p(\epsilon_{l,i})$ and $\omega = g(\phi, \epsilon)$. We apply this trick to equation 3.15 and obtain

$$\widehat{\mathcal{L}}_{VI}(\phi) = -\frac{N}{M} \sum_{i \in \mathcal{S}} \int p(\epsilon) \log p(\mathbf{y}^{(i)} | \mathbf{f}^{g(\phi, \epsilon)}(\mathbf{x}^{(i)})) d\epsilon - KL[q_\phi(\omega) || p(\omega)] \quad (3.16)$$

Applying the estimator that Kingma and Welling derived in [40] (Section 2.4) gives us the following Monte Carlo estimator:

$$\widehat{\mathcal{L}}_{MC}(\phi) = -\frac{N}{M} \sum_{i \in \mathcal{S}} \log p(\mathbf{y}^{(i)} | \mathbf{f}^{g(\phi, \epsilon)}(\mathbf{x}^{(i)})) + KL[q_\phi(\omega) || p(\omega)], \quad (3.17)$$

where $\mathbb{E}_{S, \epsilon}[\widehat{\mathcal{L}}_{MC}(\phi)] = \mathcal{L}_{VI}(\phi)$, i.e. an unbiased estimator. The log-likelihood integral can then be approximated with Monte Carlo integration to obtain an approximate posterior distribution as shown in Eq. (3.13).

Chapter 4

Variational Methods in Deep Learning

Here we presents preliminaries regarding the variational methods used in the papers. In Paper A we use dropout as a variational technique to classify time series and approximate associated uncertainty to the classification. We summarize the work of Gal [22] and outline how dropout in fact approximates variational inference. Further, we show details on how to derive the ELBO in a traditional and conditional VAE. We present the novel semi-conditional variational autoencoder (SCVAE) ELBO and show results on the MNIST data set.

Gal and Ghahramani [22, 24] showed that a neural network with arbitrary depth and non-linearities, with dropout applied before every weight layer, is mathematically equivalent to an approximation to the probabilistic deep Gaussian process. Later they extended their work [23] to convolutional neural networks, showing that CNNs regularized with dropout are equivalent VI under certain conditions (Gaussian priors and large enough hidden units). Here we outline the main steps of why and how general dropout neural networks approximate variational inference. First, we explain how dropout in neural networks operates, and secondly we provide an outline of how dropout neural networks approximate VI and thus BNNs.

4.1 Dropout Neural Networks

Stochastic regularization is the process of inducing stochastic noise in a model so that this variation efficiently functions as regularization. There are different methods by which to add stochastic noise, however, dropout [32, 60] is indisputably the most popular.

Dropout introduces noise by randomly forcing a proportion of the nodes in the model to have zero output. The nodes that are set to zero are determined by a Bernoulli distribution. During prediction, dropout is turned off, resulting in a point estimate of class probabilities. MC dropout is basically the same; however, during prediction, dropout is still turned on, randomly shutting a proportion of the nodes off. In this way, dropout generates a distribution over the model parameters by repeating the nodes sampling several times and predicting for each configuration. The process is similar to a bootstrap procedure [64]. The procedure of dropping out nodes is illustrated in Fig. 4.1.

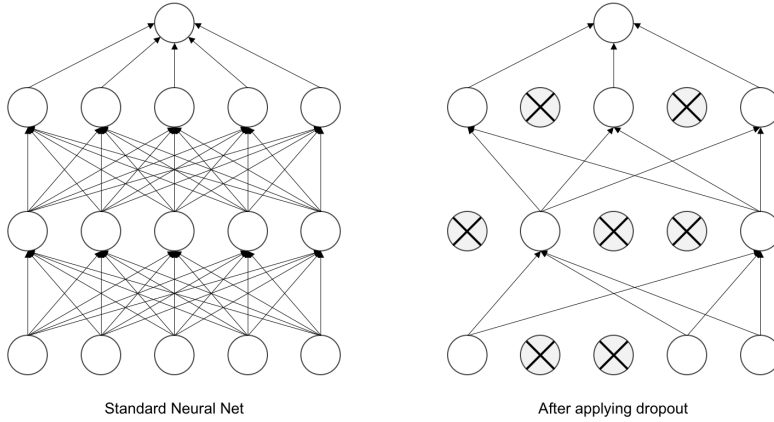


Figure 4.1: Illustration of the dropout procedure from [60].

For purpose of analysis, we want to consider dropout as a function of the parameters. We introduce two vectors \mathbf{z}_1 and \mathbf{z}_2 that have the same number of elements as the input and hidden layer, i.e. K and Q , respectively. The vector \mathbf{z}_i has elements that are either 1 or 0. Whether an element of the vector is 0 or 1 follows from a Bernoulli distribution such that \mathbf{z}_i is 1 is $0 \leq 1 - p_i \leq 1$ for $i = 1, 2$ and we write $\widehat{\mathbf{x}} = \mathbf{z}_1 \odot \mathbf{x}$. We can thus write the output of the first layer so that $\mathbf{h} = \sigma(\mathbf{W}_1 \widehat{\mathbf{x}} + \mathbf{b})$. The same procedure can be done with the hidden layer \mathbf{h} but with a percentage p_2 instead so that $\widehat{\mathbf{h}} = \mathbf{z}_2 \odot \mathbf{h}$. As for the regular neural network without dropout in Eq. (2.1) we linearly transform the output such that the output of the model becomes $\widehat{\mathbf{y}} = \widehat{\mathbf{h}} = \mathbf{W}_2 \widehat{\mathbf{h}}$. The procedure of dropping out nodes in the fashion above can be done for as many layers as necessary. During training of the network, we simply sample from the Bernoulli distribution for each vector \mathbf{z}_1 and \mathbf{z}_2 in each forward propagation and use the same samples in the backpropagation. In the next pair of forward and backward passes we sample a new distribution over \mathbf{z}_1 and \mathbf{z}_2 . We note that:

$$\begin{aligned}
 \widehat{\mathbf{y}} &= \mathbf{W}_2 \widehat{\mathbf{h}} \\
 &= \mathbf{W}_2 (\mathbf{z}_2 \odot \mathbf{h}) \\
 &= \mathbf{W}_2 (\text{diag}(\mathbf{z}_2) \mathbf{h}) \\
 &= \widehat{\mathbf{W}}_2 (\sigma(\mathbf{W}_1 (\mathbf{z}_1 \odot \mathbf{x}) + \mathbf{b})) \\
 &= \widehat{\mathbf{W}}_2 (\sigma(\mathbf{W}_1 \text{diag}(\mathbf{z}_1) \mathbf{x} + \mathbf{b})) \\
 &= \widehat{\mathbf{W}}_2 (\sigma(\widehat{\mathbf{W}}_1 \mathbf{x} + \mathbf{b})) = \mathbf{f}^{\widehat{\mathbf{W}}_1, \widehat{\mathbf{W}}_2, \mathbf{b}}
 \end{aligned} \tag{4.1}$$

where $\widehat{\mathbf{W}}_1 = \mathbf{W}_1 \text{diag}(\mathbf{z}_1)$ and $\widehat{\mathbf{W}}_2 = \mathbf{W}_2 \text{diag}(\mathbf{z}_2)$ and $\omega = \{\widehat{\mathbf{W}}_1, \widehat{\mathbf{W}}_2, \mathbf{b}\}$. We therefore can write the objective function for the dropout neural network in a similar way as in Eq. (2.4); however, here we represents mini-batches of index set S and size M :

$$\begin{aligned}
 \widehat{\mathcal{L}}_{\text{Dropout}}(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) &= \frac{1}{M} \sum_{i \in S} e^{\widehat{\mathbf{W}}_1, \widehat{\mathbf{W}}_2, \mathbf{b}}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \\
 &\quad + \lambda_1 \|\mathbf{W}_1\|^2 + \lambda_2 \|\mathbf{W}_2\|^2 + \lambda_3 \|\mathbf{b}\|^2
 \end{aligned} \tag{4.2}$$

We can express the cost function in terms of the negative log likelihood (for classification tasks) [66]; thus we can obtain the following expression for the dropout objective function

$$\widehat{\mathcal{L}}_{Dropout}(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) = -\frac{1}{M} \sum_{i \in \mathcal{S}} \log p(\mathbf{y}^{(i)} | \mathbf{f}^{\widehat{\mathbf{W}}_1, \widehat{\mathbf{W}}_2, \mathbf{b}}(\mathbf{x}^{(i)})) + \lambda_1 \|\mathbf{W}_1\|^2 + \lambda_2 \|\mathbf{W}_2\|^2 + \lambda_3 \|\mathbf{b}\|^2 \quad (4.3)$$

where $p(\mathbf{y} | \mathbf{f}^{\widehat{\mathbf{W}}_1, \widehat{\mathbf{W}}_2, \mathbf{b}}(\mathbf{x}^{(i)})) = \mathcal{N}(\mathbf{y}; \mathbf{f}^{\widehat{\mathbf{W}}_1, \widehat{\mathbf{W}}_2, \mathbf{b}}, I)$. We write the dropout operation in terms of the function

$$\widehat{\omega}_i = \{\widehat{\mathbf{W}}_1^i, \widehat{\mathbf{W}}_2^i, \mathbf{b}\} = \{\mathbf{W}_1 \text{diag}(\widehat{\boldsymbol{\epsilon}}_1^i), \mathbf{W}_2 \text{diag}(\widehat{\boldsymbol{\epsilon}}_2^i), \mathbf{b}\} = g(\psi, \widehat{\boldsymbol{\epsilon}}_i) \quad (4.4)$$

Here $\widehat{\boldsymbol{\epsilon}}_1^i \sim p(\boldsymbol{\epsilon}_1)$, and $\widehat{\boldsymbol{\epsilon}}_2^i \sim p(\boldsymbol{\epsilon}_2)$ where $1 \leq i \leq N$, and $p(\widehat{\boldsymbol{\epsilon}}_i)$ is a vector of zeros and ones, i.e. realizations from a Bernoulli distribution with a probability p_l , with same size as the columns of the \mathbf{W}_l . The index l refers to the l^{th} layer of the neural network, in this particular example $l = \{1, 2\}$. For each column of weights of the different neural network layer weights \mathbf{W}_l , there is a probability p_l that a particular column will be multiplied with zero, and thus be "dropped out". We write the dropout neural network cost function in terms of $g(\psi, \widehat{\boldsymbol{\epsilon}}_i)$ and get:

$$\widehat{\mathcal{L}}_{Dropout}(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) = -\frac{1}{M} \sum_{i \in \mathcal{S}} \log p(\mathbf{y}^{(i)} | \mathbf{f}^{g(\psi, \widehat{\boldsymbol{\epsilon}}_i)}(\mathbf{x}^{(i)})) + \lambda_1 \|\mathbf{W}_1\|^2 + \lambda_2 \|\mathbf{W}_2\|^2 + \lambda_3 \|\mathbf{b}\|^2 \quad (4.5)$$

We will later see that this is a convenient notation when comparing dropout neural networks towards the VI objective function.

4.1.1 Dropout Neural Network approximates VI

Gal [24] showed that optimizing the neural network cost function with dropout regularization is equivalent VI. Since we are using a stochastic gradient descent method, we define the derivatives of the dropout objective with respect to the parameters ψ . The derivative of the dropout objective can be expressed as:

$$\frac{\partial}{\partial \psi} \widehat{\mathcal{L}}_{Dropout}(\psi) = -\frac{1}{M} \sum_{i \in \mathcal{S}} \frac{\partial}{\partial \psi} \log p(\mathbf{y}^{(i)} | \mathbf{f}^{g(\psi, \widehat{\boldsymbol{\epsilon}}_i)}(\mathbf{x}^{(i)})) + \dots + \frac{\partial}{\partial \psi} (\lambda_1 \|\mathbf{W}_1\|^2 + \lambda_2 \|\mathbf{W}_2\|^2 + \lambda_3 \|\mathbf{b}\|^2), \quad (4.6)$$

and the derivative of the variational objective can be expressed as:

$$\frac{\partial}{\partial \phi} \widehat{\mathcal{L}}_{VI}(\phi) = -\frac{N}{M} \sum_{i \in \mathcal{S}} \frac{\partial}{\partial \phi} \log p(\mathbf{y}^{(i)} | \mathbf{f}^{g(\phi, \boldsymbol{\epsilon})}(\mathbf{x}^{(i)})) d\boldsymbol{\epsilon} + \frac{\partial}{\partial \phi} KL[q_\phi(\omega) || p(\omega)] \quad (4.7)$$

We see that these two objective functions are very similar to one another. The difference is the regularization term, and a different scaling of the log-likelihood term. We define the prior $p(\omega)$ so that the following condition holds:

$$\frac{\partial}{\partial \psi} N(\lambda_1 \|\mathbf{W}_1\|^2 + \lambda_2 \|\mathbf{W}_2\|^2 + \lambda_3 \|\mathbf{b}\|^2) = \frac{\partial}{\partial \phi} KL(q_\phi(\omega) \| p(\omega)) \quad (4.8)$$

This condition is referred to as the KL-condition. Assuming that this condition holds, we have the following relation between the objective function of dropout NN and VI:

$$\frac{\partial}{\partial \psi} \widehat{\mathcal{L}}_{Dropout}(\psi) = \frac{1}{N} \frac{\partial}{\partial \phi} \widehat{\mathcal{L}}_{VI}(\phi), \quad (4.9)$$

Gal [24] (Appendix A) proved that the KL-condition holds for a large enough number of hidden units in the case where the model priors $p(\omega)$ is a product of uncorrelated Gaussian distributions over each weight. Under the constraint mentioned above, optimizing dropout neural network is equivalent to variational inference.

4.2 Variational Autoencoders (VAE)

The main difference between a classical autoencoder Section 2.5 and the variational version is that VAE approximates probability densities instead of a point estimate of the parameters.

As earlier let $\mathbf{x}^{(i)}$ be an instance of data \mathbf{X} that arise from a continuous or discrete random variable \mathbf{x} . An unobserved continuous random variable \mathbf{z} is assumed to govern the random variable \mathbf{x} , i.e. $\mathbf{x}^{(i)} \sim p_\theta(\mathbf{x}|\mathbf{z})$. Values of $\mathbf{z}^{(i)}$ are generated from a prior distribution $p_\theta(\mathbf{z})$, that is $\mathbf{z}^{(i)} \sim p_\theta(\mathbf{z})$. Furthermore, the two probability density functions $p_\theta(\mathbf{z})$ and $\mathbf{p}_\theta(\mathbf{x}|\mathbf{z})$ are assumed parametric and differentiable with respect to the generative parameters θ and the latent representation \mathbf{z} .

We seek to find the parameters θ that in a best possible manner estimates the probability distribution $p_\theta(\mathbf{x}^{(i)}|\mathbf{z})$. From the the law of probability we have that

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \int p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z}) d\mathbf{z}. \quad (4.10)$$

Thus, we can calculate $p_\theta(\mathbf{x})$ if we know $p_\theta(\mathbf{x}|\mathbf{z})$ and $p_\theta(\mathbf{z})$. We do not know the "true" distribution $p_\theta(\mathbf{z})$; however, we can estimate it based on the data \mathbf{X} . We thus infer the probability distribution of the latent variable $p_\theta(\mathbf{z})$ conditioned on the data, i.e. $p_\theta(\mathbf{z}|\mathbf{x}^{(i)})$. The problem is that it is not trivial to approximate $p_\theta(\mathbf{z}|\mathbf{x}^{(i)})$ as well. In many contexts we turn to MC methods, however, in VAEs as the title indicates, VI is used to infer $p_\theta(\mathbf{z}|\mathbf{x}^{(i)})$. The main idea is that we can define a new probability distribution $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$ that is easy to evaluate and use this to approximate $p_\theta(\mathbf{z}|\mathbf{x}^{(i)})$. $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$ is often referred to as the recognition model. Typically $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$ is parameterized as a Gaussian, but in principle any parameterizable distribution can be used. Introduction of the recognition model is effectively turning the sampling problem into an optimization problem. We can derive the ELBO of the VAE by taking the KL-divergence between

the recognition model $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$ and the true distribution $p_\theta(\mathbf{z}|\mathbf{x}^{(i)})$

$$\begin{aligned} KL[q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) || p_\theta(\mathbf{z}|\mathbf{x}^{(i)})] &= \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}{p_\theta(\mathbf{z}|\mathbf{x}^{(i)})} \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})} \left[\log \frac{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}{p_\theta(\mathbf{z}|\mathbf{x}^{(i)})} \right] \end{aligned} \quad (4.11)$$

From Bayes theorem we have the posterior distribution that can be expressed as a product of the prior distribution $p_\theta(\mathbf{z})$ and the likelihood $p_\theta(\mathbf{x}^{(i)}|\mathbf{z})$ divided by the probability distribution of the data. Using Bayes' rule in equation 4.11 yields

$$\begin{aligned} &KL[q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) || p_\theta(\mathbf{z}|\mathbf{x}^{(i)})] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})} \left[\log q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) - \log \frac{p_\theta(\mathbf{x}^{(i)}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{x}^{(i)})} \right] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})} \left[\log q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) - (\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}) + \log p_\theta(\mathbf{z}) - \log p_\theta(\mathbf{x}^{(i)})) \right] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})} \left[\log q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) - \log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}) - \log p_\theta(\mathbf{z}) + \log p_\theta(\mathbf{x}^{(i)}) \right] \end{aligned} \quad (4.12)$$

The last term of 4.12 is thus independent of \mathbf{z} and can be moved outside the expectation so that:

$$\begin{aligned} &KL[q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) || p_\theta(\mathbf{z}|\mathbf{x}^{(i)})] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})} \left[\log q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) - \log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}) - \log p_\theta(\mathbf{z}) \right] + \log p_\theta(\mathbf{x}^{(i)}) \end{aligned} \quad (4.13)$$

Moving $\log p_\theta(\mathbf{x}^{(i)})$ to the left-hand side, we know that this will be smaller or equal to the right-hand side, since the KL-divergence is positive per definition

$$\begin{aligned} &KL[q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) || p_\theta(\mathbf{z}|\mathbf{x}^{(i)})] - \log p_\theta(\mathbf{x}^{(i)}) \leq \\ &\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})} \left[\log q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) - \log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}) - \log p_\theta(\mathbf{z}) \right] \end{aligned} \quad (4.14)$$

If we change the sign of equation 4.14 we observe that with some rearranging we can express the right-hand side as a maximum likelihood estimate regularized with a KL-divergence term:

$$\begin{aligned} &\log p_\theta(\mathbf{x}^{(i)}) - KL[q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) || p_\theta(\mathbf{z}|\mathbf{x}^{(i)})] \\ &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})} \left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}) - (\log q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) - \log p_\theta(\mathbf{z})) \right] \end{aligned} \quad (4.15)$$

$$\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})} \left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}) \right] - \mathbb{E} \left[\log q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) - \log p_\theta(\mathbf{z}) \right] \quad (4.16)$$

$$\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})} \left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}) \right] - KL[q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) || p_\theta(\mathbf{z})] \quad (4.17)$$

The right hand side is the VAE ELBO and is our objective function, i.e.

$$L_{VAE}(\theta, \phi, \mathbf{x}^{(i)}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})} \left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}) \right] - KL[q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) || p_\theta(\mathbf{z})]. \quad (4.18)$$

The first term in Eq. (4.18) is called the reconstruction term because it is a measure of the likelihood of the reconstructed data output. The K-L term can be interpreted as a regularization term, as it is a constraint on the shape of the approximate posterior.

We can choose the parameter family $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$ and $p_\theta(\mathbf{z})$. For convenience, $p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) = \mathcal{N}(\boldsymbol{\mu}^{(i)}, \text{diag}(\boldsymbol{\Sigma}^{(i)}))$. The reason for this choice is that it is possible to express $KL[q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})]$ in closed form,

$$\begin{aligned} KL[q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})] &= KL[\mathcal{N}(\boldsymbol{\mu}(\mathbf{x}^{(i)}), \boldsymbol{\Sigma}(\mathbf{x}^{(i)}))||\mathcal{N}(\mathbf{0}, \mathbf{I})] \\ &= \frac{1}{2} \left(\text{tr}(\boldsymbol{\Sigma}(\mathbf{x}^{(i)})) + \boldsymbol{\mu}(\mathbf{x}^{(i)})^T \boldsymbol{\mu}(\mathbf{x}^{(i)}) - k - \log \det(\boldsymbol{\Sigma}(\mathbf{x}^{(i)})) \right), \end{aligned} \quad (4.19)$$

where tr and \det denote the trace and the determinant of the covariance matrix. This simplifies Eq. (4.19):

$$\begin{aligned} KL[\mathcal{N}(\boldsymbol{\mu}_k(\mathbf{x}^{(i)}), \boldsymbol{\Sigma}_{k,k}(\mathbf{x}^{(i)}))||\mathcal{N}(\mathbf{0}, \mathbf{I})] &= \\ \frac{1}{2} \left(\sum_k \boldsymbol{\Sigma}_{k,k}(\mathbf{x}^{(i)}) + \sum_k \boldsymbol{\mu}_k(\mathbf{x}^{(i)})^2 - \sum_k 1 - \log \prod_k (\boldsymbol{\Sigma}_{k,k}(\mathbf{x}^{(i)})) \right) &= \\ \frac{1}{2} \left(\sum_k \boldsymbol{\Sigma}_{k,k}(\mathbf{x}^{(i)}) + \sum_k \boldsymbol{\mu}_k(\mathbf{x}^{(i)})^2 - \sum_k 1 - \sum_k \log(\boldsymbol{\Sigma}_{k,k}(\mathbf{x}^{(i)})) \right) &= \\ \frac{1}{2} \sum_k \left(\boldsymbol{\Sigma}_{k,k}(\mathbf{x}^{(i)}) + \boldsymbol{\mu}_k(\mathbf{x}^{(i)})^2 - 1 - \log(\boldsymbol{\Sigma}_{k,k}(\mathbf{x}^{(i)})) \right). \end{aligned} \quad (4.20)$$

In practice we optimize the log-variance for numerical stability.

4.3 Conditional Variational Autoencoders (CVAE)

CVAEs [59] are similar to VAEs but are different by the conditioning of the output variable on some variable, here denoted \mathbf{c} . This can be a label or some other feature of interest. Here we want to find the generative parameter θ , that maximizes the log-likelihood $\log p_\theta(\mathbf{x}^{(i)}|\mathbf{c})$. Following the same procedure as for the VAE, i.e. introducing a recognition model $q_\phi(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{c})$ (with variational parameters ϕ) as an approximation to the distribution $p_\theta(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{c})$, we can obtain the CVAE ELBO. We use the definition of the Kullback-Leibler divergence and Bayes' theorem to obtain

$$\begin{aligned} &\log p_\theta(\mathbf{x}^{(i)}|\mathbf{c}) - KL(q_\phi(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{c})||p_\theta(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{c})) \\ &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{c})} \left[-\log q_\phi(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{c}) + \log p_\theta(\mathbf{x}^{(i)}, \mathbf{z}|\mathbf{c}) \right] \\ &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{c})} \left[-\log q_\phi(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{c}) + \log p_\theta(\mathbf{z}|\mathbf{x}^{(i)}) \right] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{c})} \left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{c}, \mathbf{z}) \right] \\ &\geq KL \left[q_\phi(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{c})||p_\theta(\mathbf{z}|\mathbf{c}) \right] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{c})} \left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{c}, \mathbf{z}) \right] \\ &= \mathcal{L}_{CVAE}(\theta, \phi, \mathbf{x}^{(i)}, \mathbf{c}) \end{aligned} \quad (4.21)$$

Generative modeling of data with known labels by conditioning a certain class have been successful in multiple applications, e.g. [11, 30, 31, 34]. In the prediction phase, this allows probabilistic reconstruction of a class. Sampling randomly in the latent space together with the label information will produce different versions of this particular class. The CVAE introduces a framework that generates distribution over each class and thus permits sampling from the conditional distribution $p_\theta(\mathbf{z}|\mathbf{c})$.

4.4 Semi Conditional Variational Autoencoders (SCVAE)

In Paper B, we introduce a modification or special case of a classical CVAE for probabilistic reconstruction of data. In the paper we have focused on reconstruction related to a time-dependent process and fluid flow. Here we want to highlight that the framework can be used in a more general setting. The SCVAE aims to reconstruct data, conditioned on sparse observation of the data itself.

Let $\mathbf{w} \in \mathbb{R}^d$, $d \in \mathbb{N}$, represent some data, e.g. pixel values. Further, let $\mathcal{P} = \{p_1, \dots, p_N\}$ be a mesh that consist of N grid points p_n , $n = 1, \dots, N$. The data \mathbf{w} evaluated on \mathcal{P} can be represented as a vector $\mathbf{x}^{(i)} \in \mathbb{R}^N$,

$$\mathbf{x}^{(i)} = (w(p_1), \dots, w(p_N))^T. \quad (4.22)$$

The collection of $\mathbf{x}^{(i)}$, $i = 1, \dots, K$, constitutes the data set \mathbf{X} . Further, we assume that the information is only available only at specific points in \mathcal{P} , that is, at $\mathcal{Q} = \{q_1, \dots, q_M\} \subset \mathcal{P}$ where M is less than N . Hence, there is $\mathbf{M} = \{\mathbf{m}^{(i)} \in \mathbb{R}^M : \mathbf{m}^{(i)} = \mathbf{C} \mathbf{x}^{(i)}, \forall \mathbf{x}^{(i)} \in \mathbf{X}\}$, where $\mathbf{C} \in \mathbb{R}^{M \times N}$ is a sampling matrix. That is, \mathbf{C} is

$$(\mathbf{C})_{ij} = \begin{cases} 1, & \text{if } q_i = p_j \\ 0, & \text{otherwise} \end{cases}, \quad i = 1, \dots, M \quad j = 1, \dots, N. \quad (4.23)$$

The problem of reconstructing the $\mathbf{x}^{(i)} \in \mathbf{X}$ from $\mathbf{m}^{(i)} \in \mathbf{M}$, is presented in a schematic plot in Fig. 4.2. Here, we address the mentioned problem from a probabilistic point of

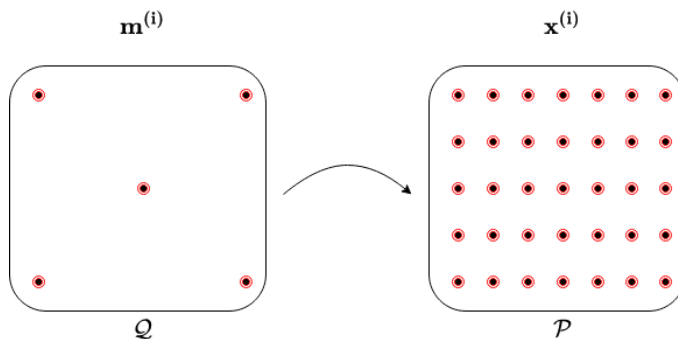


Figure 4.2: Sketch of reconstruction of $\mathbf{x}^{(i)}$ from $\mathbf{m}^{(i)}$. The dots on the right side represent the grid \mathcal{P} , and those on the left side represent the measurement locations \mathcal{Q} .

view. Let $\mathbf{x} : \mathcal{P} \rightarrow \mathbb{R}^N$ and $\mathbf{m} : \mathcal{Q} \rightarrow \mathbb{R}^M$ be two multivariate random variables associated data on \mathcal{P} and on \mathcal{Q} , respectively. Then the data sets \mathbf{X} and \mathbf{M} consist of the

realizations of \mathbf{x} and \mathbf{m} , respectively. Using \mathbf{X} and \mathbf{M} , we intend to approximate the probability distribution $p(\mathbf{x}|\mathbf{m})$. This would not only allow to predict $\mathbf{x}^{(i)}$ given $\mathbf{m}^{(i)}$, but also for a probabilistic reconstruction of $\mathbf{x}^{(i)}$. We can interpret the variation as an estimate for the uncertainty. A special version of a variational auto-encoder can be used to approximate $p(\mathbf{x}|\mathbf{m})$.

In the SCVAE we use the measurement $\mathbf{m}^{(i)}$ as the conditioning to estimate the probability function $p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{m})$. Since $\mathbf{m}^{(i)} = C\mathbf{x}^{(i)}$, where C is non-stochastic we can simplify the encoder and recognition model to

$$p_\theta(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{m}^{(i)}) = p_\theta(\mathbf{z}|\mathbf{x}^{(i)}) \quad \text{and} \quad q_\phi(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{m}^{(i)}) = q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) \quad (4.24)$$

Hence, the CVAE ELBO can be updated and we can obtain the SCVAE ELBO as follows

$$\begin{aligned} \mathcal{L}_{SCVAE}(\theta, \phi, \mathbf{x}^{(i)}, \mathbf{m}^{(i)}) = & KL \left[q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) || p_\theta(\mathbf{z}|\mathbf{m}^{(i)}) \right] \\ & + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})} \left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{m}^{(i)}, \mathbf{z}) \right] \end{aligned} \quad (4.25)$$

Maximizing the SCVAE ELBO Eq. (4.25), will maximize $\log p_\theta(\mathbf{x}^{(i)}|\mathbf{m}^{(i)})$, that is reconstructing $\mathbf{x}^{(i)}$ given the measurements $\mathbf{m}^{(i)}$. The SCVAE structure allows for a simplification of the auto-encoder structure. This means that we do not need to consider the measurements in the encoding phase. With assumptions that $p_\theta(\mathbf{z}|\mathbf{m})$ is normally distributed and that $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$ is Gaussian (as in Section 4.2), we can estimate the KL-term as given in Eq. (4.20). The reconstruction term can be estimated with a MC estimator and the reparametrization trick as [40]

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})} \left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}, \mathbf{m}^{(i)}) \right] \approx \frac{1}{L} \sum_{l=1}^L \log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)}, \mathbf{m}^{(i)}), \quad (4.26)$$

$$\text{where } \mathbf{z}^{(i,l)} = g_\phi(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)}, \mathbf{m}^{(i)}), \quad \boldsymbol{\epsilon}^l \sim p(\boldsymbol{\epsilon}). \quad (4.27)$$

Here $\boldsymbol{\epsilon}^l$ is an auxiliary (noise) variable with independent marginal $p(\boldsymbol{\epsilon})$, L is the number of samples and $g_\phi(\cdot)$ is a differentiable transformation of $\boldsymbol{\epsilon}$, parametrized by ϕ , for details see [40].

4.4.1 SCVAE on MNIST Dataset

We illustrate the SCAE performance on the classical MNIST (Modified National Institute of Standards and Technology) dataset. The MNIST data set consists of 60000 digital grey scale images of handwritten digits with dimension (28×28) , that are used for training and validation. The test dataset consists of 10000 images, so that in total there are 70000 images. The 60000 instances were split randomly, such that 30% constitute a validation data set, and the remainder the training data set. This implies training data and a validation dataset consisting of 42000 and 18000 instances, respectively.

We generate the data set \mathbf{M} as described in Eq. (4.23) by uniformly withdraw observations from \mathbf{X} . We create four different data sets, with 64, 36, 25 and 16 observations

in each image. We normalize the pixel values in both \mathbf{M} and \mathbf{X} to improve the conditioning of the optimization problem for each of the test, train and validation data sets. Four different models are trained, dependent on the number of measurements in the images. As a dimension reduction technique we use two convolutional layers with strides and kernel sizes of two. The reparametrization trick is applied in the decoder after the convolutional layers. The output of the reparametrization operation produces the output of the encoder, i.e. the latent representation \mathbf{z} with latent dimension of two. The decoder takes the measurements $\mathbf{m}^{(i)}$ and the latent representation \mathbf{z} as input. The two inputs are concatenated and reshaped to the original structure by using transposed convolutional layers with strides and kernel sizes two.

For optimization of the models we use mini-batch optimization with batch sizes of 64 and the RMSProp algorithm. We run all models to 60 epoch and predict on random samples in the test data set to illustrate the reconstruction and associated uncertainty. We want to emphasize that in this operation, we have not spent much time on optimizing hyperparameters. With a more structured optimization of the hyperparameters, results could be improved.

Fig. 4.3 shows column-wise the true label, the observations, the mean prediction and prediction with uniform sampling over \mathbf{z} . From first to last row in Fig. 4.3 results are shown with 64, 36, 25 and 16 pixels as measurements, respectively. With 64, 36 and 25 pixels as measurements, we observe that the mean prediction is quite good for this particular instance in the test data. The last row shows reconstruction with only 16 measurements. The reconstruction can be mistaken as an eight instead of a five.

We input the test data \mathbf{X} to the encoder, and predict the latent distribution for all instances in the test dataset. Fig. 4.4 shows the prediction for the different models. We observe that the latent representation with 64 and 36 observations have a distribution that better resembles a 2D normal distribution, than to the models with 25 and 16 observations. To compare the different models, we calculate the relative mean L_2 -error of the reconstructed images. That is,

$$\mathcal{E} = \frac{1}{n} \sum_{i=1}^n \frac{\|\widehat{\mathbf{x}}^{(i)} - \mathbf{x}^{(i)}\|_2}{\|\mathbf{x}^{(i)}\|_2} \quad (4.28)$$

where $\widehat{\mathbf{x}}^{(i)}$ is the mean prediction. From Fig. 4.5 we observe that the relative L_2 error increases with a corresponding decrease in the number of measurements. With 64, 36, 25 and 16 measurements, the relative L_2 -error (See, Eq. (4.28)) for all samples in the test data is 0.123, 0.241, 0.403 and 0.516, respectively. The spread or variance of the error also increases with decreasing measurements, that is the standard deviations are 0.085, 0.144, 0.184 and 0.241, for the different models. This is according to expectations, as the model is conditioned on less information with a decreasing number of observations.

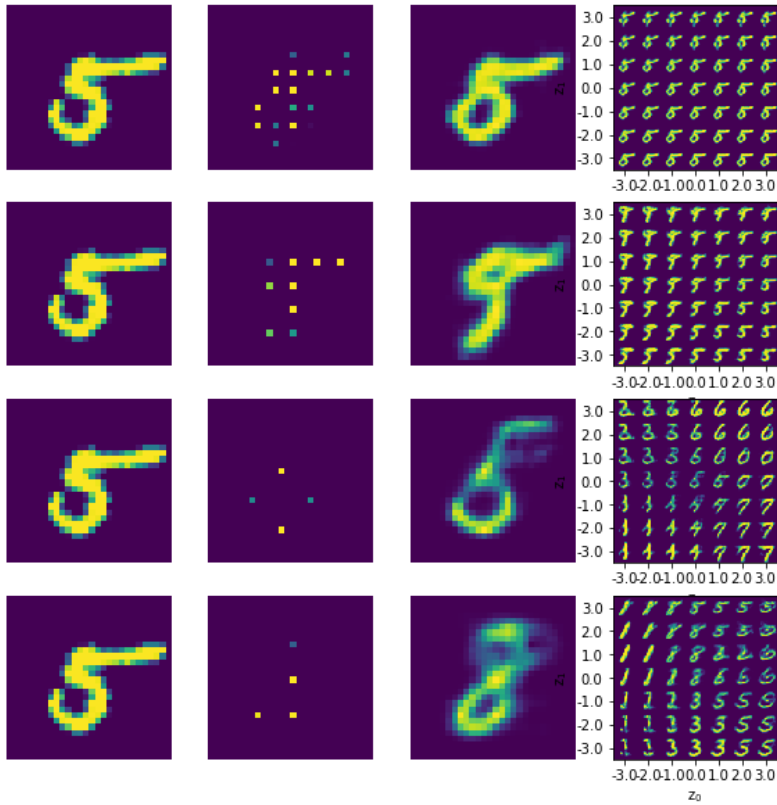


Figure 4.3: **First Column:** The true label/instance. **Second Column:** From first to last row; 64, 36, 25 and 16 uniformly sampled observations used as conditioning in the SCVAE **Third Column:** Mean prediction, i.e. $\mathbf{z} = [0, 0]$ **Fourth Column:** Uniformly sampling over the latent representation with predictions.

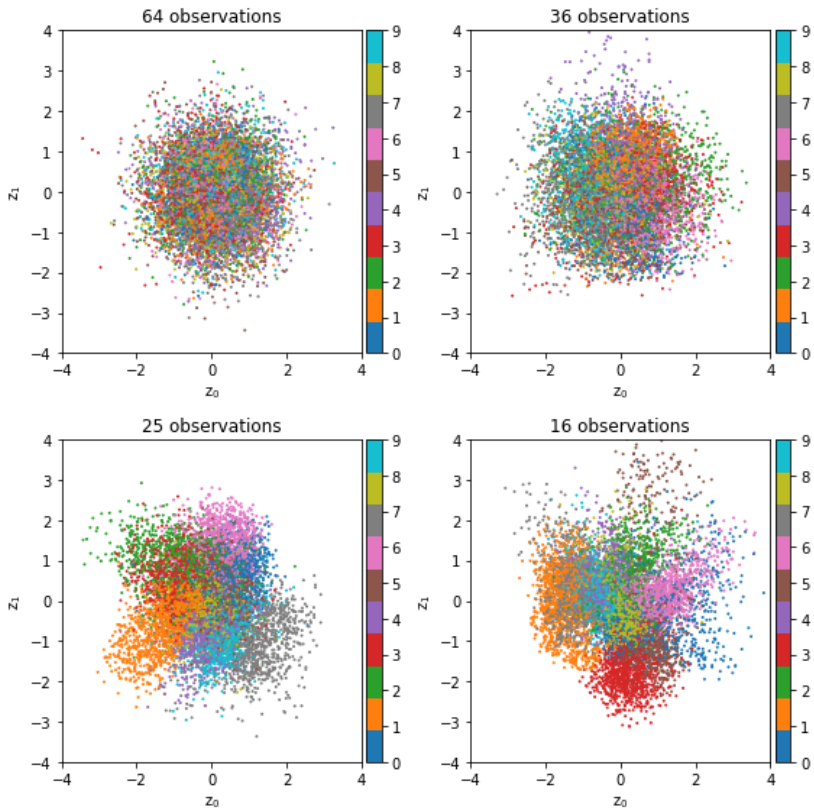


Figure 4.4: The latent representation for the test data set for the different model, i.e. trained with 64, 36, 25, and 16 fixed observations. The different colours show the different numbers in the data set.

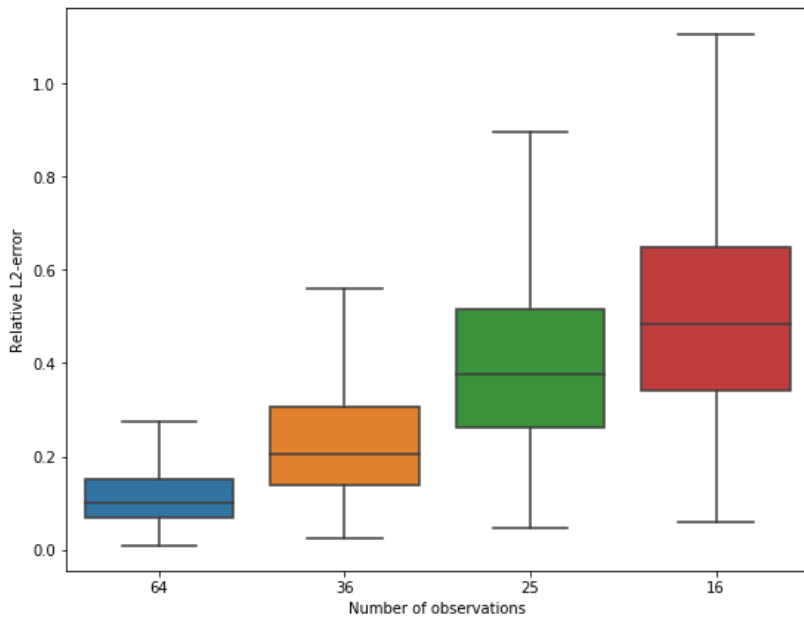


Figure 4.5: Relative L2 error (See Eq. (4.28)) for the models trained with 64, 36, 25 and 16 uniform fixed observations.

Chapter 5

Introduction to the papers

Paper A: Binary Time Series Classification with Bayesian Convolutional Neural Networks when Monitoring for Marine Gas Discharges

*Kristian Gundersen**, Anna Oleynik, Nello Blaser, Guttorm Alendal (2020), *Algorithms*, volume(13/145)

Paper A investigates detection of abnormal time series and CO₂ levels on the seafloor from an unwanted discharge from CCS sites that the leak into the marine environment from subsurface storage sites. In a marine monitoring situation, expensive cruises to confirm potential leakages has to be minimized. Traditional time series classification techniques and deep learning approaches to the problem usually produce a deterministic outcome if there is a leak or non leak. In Paper A we use a BNN for binary classification of time series. The predicted labels represents either a leak or no-leak situation. We use a BCNN for the classification, that is a classical CNN where we have applied the MC dropout technique for uncertainty quantification. This ensures probabilistic predictions of what class the time series belongs to, and provides valuable information for decision-makers. We present an algorithm that uses the posterior predictive distribution from the BCNN framework to make informed and optimal decisions under a specific cost attributed to the different actions and outcomes.

Paper B: Semi Conditional Variational Auto-Encoder for Flow Reconstruction and Uncertainty Quantification from Limited Observations

*Kristian Gundersen**, Anna Oleynik, Nello Blaser, Guttorm Alendal, *Submitted to Physics of Fluids*, August 2020, published on arxiv.org

Paper B addresses the issue of reconstruction of flow fields based on sparse observations with associated uncertainty quantification. We use a variational auto-encoder to solve the problem. The approach is tested on two different data sets, flow around a 2D cylinder and currents from the ocean model BOM. The results shows good performance in comparison with the more traditional Gappy Proper Orthogonal Decomposition approach. This method can be used to quantify the impact of a leakage in a probabilistic manner.

Paper C: A Variational Auto-encoder for Reservoir Monitoring

*Kristian Gundersen**, Seyyed Hossini, Anna Oleynik, Guttorm Alendal, *To be submitted to Machine Learning, published on arxiv.org*

Paper C focus on subsurface monitoring through observations wells in the AZMI. We simulate leakages in a CO₂ storage reservoir and record the impact on the pressure in the AZMI. A model that both reconstructs the pressure fields and classifies the flux of the leakage is presented. The approach quantifies not only the uncertainty of the reconstructed pressure, but also the uncertainty of classification of the flux of the leakages. In this case we optimize two tasks simultaneously, hence, the proposed model is a so-called multitask learning (MTL) framework. With a full dense pressure field, it is possible to estimate the location of the leakage. The estimation of the flux makes it possible to draw conclusions about the severity of the event.

Chapter 6

Contribution and Outlook

Over the last decade there have been several projects, e.g., QICS (Quantifying and Monitoring Potential Ecosystem Impacts of Geological Carbon Storage), ETI MMV (Energy Technologies Institute Measurement, Monitoring and Verification of CO₂ Storage), and STEMM-CCS (Strategies for Environmental Monitoring of Marine CCS), dedicated to monitoring of CCS sites. One of the remaining challenges in implementation of CCS monitoring program is how to treat the large amounts of data generated by such a program. A recently published paper by Dean et. al. [16], pinpoints the need for research for dealing with missed/false alerts due to large variations in the background signal as well as methods for real time decision making are needed. One of the advantages with deep learning approaches is that they can detect sophisticated patterns masked by the natural variability, thus increasing the detectability over more conventional methods. The proposed algorithm in Paper A is an approach that deals with the false/positive issue through uncertainty quantification, and use this information for making optimal decision. An potential extension of the work presented in Paper A is to include multivariate time series data, e.g. pH, alkalinity, pressure or temperature to improve the classification. Another possible extension is to use transfer learning, i.e. first train the classifier on model data and then fix the weights of the first layers and train the last layers on limited in-situ data. In the experiment presented in Paper A, the BCNN-model is optimized with data based on a limited number of simulations for a limited time period. This biases the BCNN-model towards the simulated conditions. The model can still be used in a general scenario, but its predictive power will decrease. That's why more simulations with different forcing, leak locations and fluxes are necessary to generate a predictive model that is more robust.

One of the major issues in design of monitoring programs is ascertaining where to place sensors. In combination with an advection-diffusion model, we can use the SC-VAE presented in Paper B to create statistical sound velocity fields, that in turn can be used to create statistical sound footprints of marine leakages. An MC estimator can be developed for this purpose, that is, we can use the advection-diffusion model and run it multiple times and integrate over the possible release locations, the velocity field distribution and leakage flux. The numerical ocean models are well suited to generate velocity fields, they rely on multiple inputs, including weather conditions. Even if these inputs are readily available, which frequently is not the case, data assimilation methods must be used to produce reliable results and avoid a systematic bias. Training the

SCVAE model might be time consuming; however, the prediction is not. This allows for fast production of probabilistic velocity fields without the need of running computationally demanding ocean models. We want to emphasize that the SCVAE could be used in a wide range of applications, where the target is to go from sparse observations to representation over an entire domain, and where uncertainty quantification is important. In Section 4.4 we showed that the method can be applied in computer vision as well. A natural extension of the SCVAE is to add time-dependency to the model. This could be done by introducing a Markov assumption. Then instead of approximating $p_{\theta}(\mathbf{x}^{(i)}|\mathbf{m}^{(i)})$ we approximate $p_{\theta}(\mathbf{x}^{(i)}|\mathbf{m}^{(i)}, \mathbf{x}^{(i-1)})$.

Governments and legislation demand that operators of CCS sites have proper subsurface monitoring program. The MTL framework proposed in Paper C can be a valuable tool to quickly obtain information about where the leak is located and the severity of it. If CO₂ leaks to the subsurface environment and subsequently to the atmosphere, operators will be held accountable. The MTL-model can be used to determine the environmental and financial consequences of a leakage. The proposed MTL-model can be used to optimizing the placement of the AZMI-wells. Training multiple MTL-models with different locations of the well placement, and choose the model that minimizes the error of both the quantification of flux and best possible recreate the pressure may serve this purpose. In Paper C we simulated leaks with the same porosity and permeability for all leakage scenarios. A natural extension of our work is to alter the porosity and permeability during generation of input to the model, since these parameters are uncertain. Ideally, many more simulations should be included in a in-situ monitoring situation.

Through out the papers we have relied on output from models (e.g. ocean models and reservoir simulators) as input for our data driven methods. These models are inherently inaccurate, as they depend on numerical schemes, discretization, and assumptions related to the forcing of the model. It is possible to hindcast or history match with historical observations to improve the model's performance. Methods such as EnKF can be used to improve the fit between the model and measurements. One of the major problems with model data that it is smoother than in-situ data. Prediction using in-situ data, that is trained on data from models that are not hindcast or history matched may yield poor results. We have not used history matched or hindcasted data, which would be needed if the model to be used for site monitoring.

To use data driven methods and ANNs to explain physical phenomena can be controversial. The crown argument against artificial neural network is that they are so called black boxes, meaning that models is highly complex, and an understanding the mechanism behind the model prediction is elusive. ANN yields exceptional results in many cases but at the expense of low interpretability of the model and outcome. Improvement of interpretability of the deep learning model is currently a field of research.

Bibliography

- [1] Al Mamun S. M. A., Lu C., and Jayaraman B. Extreme learning machines as encoders for sparse reconstruction. *Fluids*, 3(4), 2018. 1.2
- [2] International Energy Agency. Global energy & co₂ status report. Technical report, IEA, 2018. 1.1
- [3] Guttorm Alendal, Jarle Berntsen, Elisabeth Engum, Gunnar K Furnes, Gudmund Kleiven, and Lars I Eide. Influence from ocean weather on near seabed currents and events at ormen lange. *Marine and Petroleum geology*, 22(1-2):21–31, 2005. 1.2
- [4] Guttorm Alendal and Helge Drange. Two-phase, near-field modeling of purposefully released co₂ in the ocean. *Journal of Geophysical Research: Oceans*, 106(C1):1085–1096, 2001. 1.2
- [5] Jarle Berntsen. Users guide for a modesplit σ -coordinate numerical ocean model. *Department of Applied Mathematics, University of Bergen, Tech. Rep.*, 135:48, 2000. 1.2
- [6] Jerry Blackford, Yuri Artioli, James Clark, and Lee de Mora. Monitoring of offshore geological carbon storage integrity: Implications of natural variability in the marine system and the assessment of anomaly detection criteria. *International Journal of Greenhouse Gas Control*, 64:99–112, 2017. 1.2
- [7] Jerry Blackford, Jonathan M Bull, Melis Cevatoglu, Douglas Connelly, Chris Hauton, Rachael H James, Anna Lichtschlag, Henrik Stahl, Steve Widdicombe, and Ian C Wright. Marine baseline and monitoring strategies for Carbon Dioxide Capture and Storage (CCS). *International Journal of Greenhouse Gas Control*, 38:221–229, July 2015. 1.2
- [8] Jerry C. Blackford, S Widdicombe, D Lowe, and B Chen. Environmental risks and performance assessment of carbon dioxide (CO₂) leakage in marine ecosystems. In *Developments and Innovation in Carbon Dioxide (CO₂) Capture and Storage Technology, Volume 2 - Carbon Dioxide (CO₂) Storage and Utilisation.*, pages 344–373. Woodhead Publishing Limited, September 2010. 1.2
- [9] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017. 3.3

- [10] Helle Botnen, AM. Omar, I. Thorseth, T. Johannessen, and G. Alendal. The effect of submarine CO₂ vents on seawater: implications for detection of subsea Carbon sequestration leakage. *Limnology and Oceanography*, 60(2), 2015. 1.2
- [11] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015. 4.3
- [12] Amanda D Boyd, Yue Liu, Jennie C Stephens, Elizabeth J Wilson, Melisa Pollak, Tarla Rai Peterson, Edna Einsiedel, and James Meadowcroft. Controversy in technology innovation: Contrasting media and expert risk perceptions of the alleged leakage at the weyburn carbon dioxide storage demonstration project. *International Journal of Greenhouse Gas Control*, 14:259–269, 2013. 1.2
- [13] Peter G Brewer, Baixin Chen, Robert Warzinki, Arthur Baggeroer, Edward T Peltzer, Rachel M Dunk, and Peter Walz. Three-dimensional acoustic monitoring and modeling of a deep-sea co₂ droplet cloud. *Geophysical research letters*, 33(23), 2006. 1.2
- [14] Changsheng Chen. *An Unstructured-grid, Finite-volume Community Ocean Model: FVCOM User Manual*. Sea Grant College Program, Massachusetts Institute of Technology, 2012. 1.2
- [15] Haskell B Curry. The method of steepest descent for non-linear minimization problems. *Quarterly of Applied Mathematics*, 2(3):258–261, 1944. 2.2
- [16] Marcella Dean, Jerry Blackford, Douglas Connelly, and Rob Hines. Insights and guidance for offshore co₂ storage monitoring based on the qics, eti mmv, and stemm-ccs projects. *International Journal of Greenhouse Gas Control*, 100:103120, 2020. 1.2, 6
- [17] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural safety*, 31(2):105–112, 2009. 3.1
- [18] Elianny Domínguez-Tejo, Graciela Metternicht, Emma Johnston, and Luke Hedge. Marine Spatial Planning advancing the Ecosystem-Based Approach to coastal zone management: A review. *Marine Policy*, 72:115–130, October 2016. 1.2
- [19] Helge Drange, Guttorm Alendal, and Ola M Johannessen. Ocean release of fossil fuel co₂: A case study. *Geophysical Research Letters*, 28(13):2637–2640, 2001. 1.2
- [20] Geir Evensen. *Data assimilation: the ensemble Kalman filter*. Springer Science & Business Media, 2009. 1.2
- [21] Plenary Julia First. Global warming of 1.5 c an ipcc special report on the impacts of global warming of 1.5 c above pre-industrial levels and related global greenhouse gas emission pathways, in the context of strengthening the global response to the threat of climate change. *Sustainable Development, and Efforts to Eradicate Poverty*. <https://www.ipcc.ch/sr15/>. Accessed, 1, 2019. 1.1

- [22] Yarin Gal. *Uncertainty in deep learning*. PhD thesis, University of Cambridge, 2016. 2, 3, 4
- [23] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with Bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015. 4
- [24] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016. 4, 4.1.1, 4.1.1
- [25] Jon Gibbins and Hannah Chalmers. Carbon capture and storage. *Energy policy*, 36(12):4317–4322, 2008. 1.1
- [26] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 6.2.2.3 *Softmax Units for Multinoulli Output Distributions*. MIT press, 2016. 2.1
- [27] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Chapter 9 Convolutional Networks Section 9.2 Motivation*. MIT press, 2016. 2.4
- [28] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. 2.5
- [29] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016. 2
- [30] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015. 4.3
- [31] Aditya Grover and Stefano Ermon. Uncertainty autoencoders: Learning compressed representations via variational information maximization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2514–2524, 2019. 4.3
- [32] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012. 4.1
- [33] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989. 2.1
- [34] Oleg Ivanov, Michael Figurnov, and Dmitry Vetrov. Variational autoencoder with arbitrary conditioning. *arXiv preprint arXiv:1806.02382*, 2018. 4.3
- [35] DG Jones, SE Beaubien, JC Blackford, EM Foekema, Julie Lions, C De Vittor, JM West, S Widdicombe, C Hauton, and AM Queirós. Developments since 2005 in understanding potential environmental impacts of co2 leakage from geological storage. *International Journal of Greenhouse Gas Control*, 40:350–377, 2015. 1.2

- [36] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999. 3.1
- [37] Abbas Khosravi, Saeid Nahavandi, Doug Creighton, and Amir F Atiya. Comprehensive review of neural network-based prediction intervals and new advances. *IEEE Transactions on neural networks*, 22(9):1341–1356, 2011. 3
- [38] Seunghee Kim and Seyyed Abolfazl Hosseini. Above-zone pressure monitoring and geomechanical analyses for a field-scale co2 injection project in cranfield, ms. *Greenhouse Gases: Science and Technology*, 4(1):81–98, 2014. 1.2
- [39] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2.2
- [40] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2.5, 3.3.1, 3.3.1, 3.3.1, 4.4, 4.4
- [41] Werner Krauth. Introduction to monte carlo algorithms. In *Advances in Computer Simulation*, pages 1–35. Springer, 1998. 3.2
- [42] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951. 3.3
- [43] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017. 3
- [44] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2.4, 2.4
- [45] Yingzhen Li, José Miguel Hernández-Lobato, and Richard E Turner. Stochastic expectation propagation. In *Advances in neural information processing systems*, pages 2323–2331, 2015. 3.1
- [46] Timothy Meckel, Susan Hovorka, et al. Above-zone pressure monitoring as a surveillance tool for carbon-sequestration projects. In *SPE International Conference on CO2 Capture, Storage, and Utilization*. Society of Petroleum Engineers, 2010. 1.3
- [47] Stefano Menegon, Daniel Depellegrin, Giulio Farella, Alessandro Sarretta, Chiara Venier, and Andrea Barbanti. Addressing cumulative effects, maritime conflicts and ecosystem services threats through msp-oriented geospatial webtools. *Ocean & Coastal Management*, 163:417–436, 2018. 1.2
- [48] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. 2.1
- [49] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012. 3.1

- [50] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011. 3.2
- [51] Geir K Nilsen, Antonella Z Munthe-Kaas, Hans J Skaug, and Morten Brun. On the delta method for uncertainty approximation in deep learning. *arXiv preprint arXiv:1912.00832*, 2019. 3
- [52] Ryan RP Noble, Linda Stalker, Steven A Wakelin, Bobby Pejic, Matthew I Leybourne, Allison L Hortle, and Karsten Michael. Biological monitoring for carbon capture and storage—a review and potential future developments. *International Journal of Greenhouse Gas Control*, 10:520–535, 2012. 1.2
- [53] Oil and Energy Department of Norway. Melding til stortinget 33: Langskip fangst og lagring av co2. Technical report, Oil and Energy Department of Norway, 2020/2019. 1.1
- [54] C. M. Oldenburg and J. L. Lewicki. On leakage and seepage of CO₂ from geologic storage sites into surface water. *Environmental Geology*, 50(5):691–705, July 2006. 1.2
- [55] Anna Oleynik, Maribel I García-Ibáñez, Nello Blaser, Abdirahman Omar, and Guttorm Alendal. Optimal sensors placement for detecting co2 discharges from unknown locations on the seafloor. *International Journal of Greenhouse Gas Control*, 95:102951, 2020. 1.2
- [56] Stephen A Rackley. *Carbon capture and storage*. Butterworth-Heinemann, 2017. 1.1
- [57] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951. 2.2, 3.3.1
- [58] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988. 2.2
- [59] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491, 2015. 4.3
- [60] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 3, 4.1, 4.1
- [61] Chris E Strickland, Vince R Vermeul, Alain Bonneville, E Charlotte Sullivan, Tim C Johnson, Frank A Spane, Tyler J Gilmore, et al. Geophysical monitoring methods evaluation for the futuregen 2.0 project. *Energy Procedia*, 63(C), 2014. 1.2
- [62] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013. 2.2

- [63] M.S.P. IOC-UNESCO the Intergovernmental Oceanographic Commission of UNESCO. Marine spatial planning programme. <http://msp.ioc-unesco.org>. 1.2
- [64] Robert J Tibshirani and Bradley Efron. An introduction to the bootstrap. *Mono-graphs on statistics and applied probability*, 57:1–436, 1993. 4.1
- [65] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012. 2.2
- [66] Naftali Tishby, Esther Levin, and Sara A Solla. Consistent inference of probabilities in layered networks: Predictions and generalization. In *International Joint Conference on Neural Networks*, volume 2, pages 403–409, 1989. 2.2, 4.1
- [67] Jay M Ver Hoef. Who invented the delta method? *The American Statistician*, 66(2):124–127, 2012. 3
- [68] Lisa Vielstädte, Jens Karstens, Matthias Haeckel, Mark Schmidt, Peter Linke, Susan Reimann, Volker Liebetrau, Daniel F McGinnis, and Klaus Wallmann. Quantification of methane emissions at abandoned gas wells in the central north sea. *Marine and Petroleum Geology*, 68:848–860, 2015. 1.2
- [69] Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008. 3.1
- [70] G Wegener, M. Shovitri, K Knittel, H Niemann, M Hovland, and A Boetius. Biogeochemical processes and microbial diversity of the Gullfaks and Tommeliten methane seeps (Northern North Sea). *Biogeosciences Discussions*, 5(1):971–1015, February 2008. 1.2
- [71] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011. 3.1

Part II
Included Papers



Paper A

Binary Time Series Classification with Bayesian Convolutional Neural Networks when Monitoring for Marine Gas Discharges

Kristian Gundersen, Nello Blaser, Anna Oleynik, Guttorm Alendal
MDPI, Algorithms, volume/number (2020)

Article

Binary Time Series Classification with Bayesian Convolutional Neural Networks When Monitoring for Marine Gas Discharges

Kristian Gundersen ^{1,*}, Guttorm Alendal ¹ , Anna Oleynik ¹ and Nello Blaser ² 

¹ Department of Mathematics, University of Bergen, 5020 Bergen, Norway; Guttorm.Alendal@uib.no (G.A.); Anna.Oleynik@uib.no (A.O.)

² Department of Informatics, University of Bergen, 5020 Bergen, Norway; Nello.Blaser@uib.no

* Correspondence: Kristian.Gundersen@uib.no

Received: 27 February 2020; Accepted: 12 June 2020; Published: 19 June 2020



Abstract: The world's oceans are under stress from climate change, acidification and other human activities, and the UN has declared 2021–2030 as the decade for marine science. To monitor the marine waters, with the purpose of detecting discharges of tracers from unknown locations, large areas will need to be covered with limited resources. To increase the detectability of marine gas seepage we propose a deep probabilistic learning algorithm, a Bayesian Convolutional Neural Network (BCNN), to classify time series of measurements. The BCNN will classify time series to belong to a leak/no-leak situation, including classification uncertainty. The latter is important for decision makers who must decide to initiate costly confirmation surveys and, hence, would like to avoid false positives. Results from a transport model are used for the learning process of the BCNN and the task is to distinguish the signal from a leak hidden within the natural variability. We show that the BCNN classifies time series arising from leaks with high accuracy and estimates its associated uncertainty. We combine the output of the BCNN model, the posterior predictive distribution, with a Bayesian decision rule showcasing how the framework can be used in practice to make optimal decisions based on a given cost function.

Keywords: deep learning; Bayesian convolutional neural network; uncertainty quantification; time series classification; CO₂-leak detection

1. Introduction

The world's oceans are under tremendous stress from global warming, ocean acidification and other human activities [1], and the UN has declared 2021–2030 as the ocean decade (<https://en.unesco.org/ocean-decade>). Monitoring the marine environment is a part of the ecosystem-based Marine Spatial Planning initiative by the IOC [2] and Life Under Water is number 14 of the UN's Sustainable Development Goals.

The aim here is to study how the use of machine-learning techniques, combined with physical modeling, can assist in designing and operating a marine environmental monitoring program. The purpose of monitoring is to detect tracer discharges from an unknown location. Examples are accidental release of radioactive, biological, or chemical substances from industrial complexes, e.g., organic waste from fish farms in Norwegian fjords [3] and other contaminants that might have adverse effects on marine ecosystems [4].

As a case study, we use the monitoring of areas in which large amounts of CO₂ are stored in geological formations deep underneath the seafloor. Such storage is a part of the Carbon Capture and Storage (CCS) technology and, according to the International Energy Agency and The

Intergovernmental Panel on Climate Change, will be a key factor to reach the below -1.5 °C goal and should account for 14% of the total CO₂ reduction [5,6]. Due to the large amount of CO₂ to be stored, and as a precaution, the marine environment will have to be monitored for indications of a leak through the seafloor to compile with regulations (<https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32009L0031&from=EN>) [7–9].

A challenge for detecting CO₂ seeps to marine waters is that CO₂ is naturally present in marine environments and the concentration is highly variable due to local transport by water masses, uptake of atmospheric CO₂, temperature, biological activity, geochemistry of the sediments and other factors. Therefore, a CO₂ seep signal can be hidden within natural variability. The purpose here is to classify noisy time series into two classes: leak vs no-leak.

Time series classification (TSC), or distinguishing of ordered sequences, is a classical problem within the field of data mining [10,11]. The problem has been tackled with numerous different approaches; see for instance Bagnall et al. [12].

Distance-based methods use a measure of distance, or similarity, between time series. They then combine them with a distance-based classifier. Dynamic Time Warping (DTW) and Euclidean distance are typical examples of metrics between time series. DTW seems to be the most successful distance-based method, as it allows for perturbations, shifts, variations and in the temporal domain [13,14].

Feature-based methods extract features from the time series and use traditional classification methods on the extracted features [12]. Traditional signal-processing techniques, using various transforms, e.g., Fast Fourier Transform or Discrete Wavelet Transform, are often used as preprocessing steps to generate features for traditional classifiers.

Model-based TSC uses time series generated by some underlying process model, and new time series can be assigned to the model class that fits best. Typical models used for model-based time series classification are auto-regressive models [15] and hidden Markov models [16].

There are also many other techniques, such as Gaussian processes [17] and functional data analysis [18,19] that can be successfully applied to the TSC problem.

According to Bagnall et al. [12], the state-of-the-art TSC algorithms are the Collective of Transformation Based Ensembles (COTE) [20] and Dynamic Time Warping (DTW) [13,14] in combination with some classifier, e.g., *k*-nearest-neighbor or decision tree. More recently, COTE has been extended to use a hierarchical vote system, resulting in the HIVE-COTE algorithm, a significant improvement over the COTE algorithm [21]. HIVE-COTE is a hierarchical method combining different classifiers into ensembles to increase the performance of the classification capabilities. It combines 37 different classifiers that use all the above-mentioned techniques, including frequency, and shapelet transformation domains. HIVE-COTE is currently the classifier that performs best on the UCR-datasets [12], and it is considered state of the art in TSC. The major drawback with both COTE and DTW methods is the high computational cost.

Lately, the hegemony of COTE and DTW has been challenged by several efforts of using artificial neural networks for TSC [22]. For example Zheng et al. [23] applied Multi-Channel Deep Convolutional Neural Networks (MC-CNN) on data for human activity as well as on congestive hearing failure data. They found that MC-CNN is more efficient, and competitive in accuracy, compared to state-of-the-art traditional TSC algorithms (1-NN DTW). A fully Convolutional Neural Network (CNN), deep multilayer perceptions network (Dense Neural Network, DNN) and a deep Residual Neural Network architecture was tested in a univariate TSC setting in [24]. They found that both their fully connected Convolutional Neural Network and the deep Residual Neural Network architectures achieve better performance compared to other state-of-the-art approaches.

Both Recurrent Neural Networks (RNN) [25] and Convolutional Neural Networks (CNN) for TSC are showing state-of-the-art performance, outperforming other techniques on some datasets, but not on others [22–24]. Due to their nature, RNNs are a natural choice when dealing with time series; however, one of their drawbacks is that they use more time on optimization. In a review of

TSC methods, Fawaz et al. [22] found that CNNs outperform RNNs not only in training time, but also in accuracy.

An important issue that many TSC techniques cannot achieve is to quantify prediction uncertainty. One way to overcome this limitation, and retain state-of-the-art predictive power, is to use Bayesian Neural Networks [26], e.g., Bayesian Convolutional Neural Networks (BCNNs) or Bayesian Recurrent Neural Networks (BRNNs). These have the same advantages as the standard neural networks, but have the additional benefit of providing posterior predictive distributions.

When dealing with a binary classification problem, such as the leak vs. no-leak classification used here, the output from a Bayesian Neural Network is a probability estimate, or level of uncertainty, of the class that a given time series belongs to. This uncertainty is important information when making decisions based on the classification, such as to mobilize a costly confirmation and localization survey [27].

The major drawback with classical Bayesian Neural Networks is their failure to up-scale to large data sets. Gal and Ghahramani [28] have recently proposed to use Bernoulli dropout during both the training and the testing stages. This can be viewed as an approximate variational inference technique [28]. Blundell et al. [29] presented an algorithm called *Bayes by backprop* and showed that it can efficiently estimate the distributions over the model parameters. Shridhar et al. [30] applied the Bayes by backprop algorithm in different CNNs and for different data sets, and compared it with the Gal and Ghahramani approach. They found that the two approaches are comparable. The simple and applicable nature of the method by Gal and Ghahramani has made it popular in a wide range of applications where quantification of uncertainty is important, e.g., [31–33].

In their review of the status and future of deep-learning approaches in oceanography, Malde et al. [34] argues that deep learning is still an infant method within marine science. Neural network models have been applied to various environmental data, e.g., [35,36], and there have been some efforts regarding classification of environmental time series, e.g., [37,38]. To our knowledge, using a probabilistic deep-learning approach, such as BCNN, has yet to be explored on the environmental time series. This is the motivation for the present work.

We use Gal and Ghahramani's BCNN on the classical statistical problem of TSC and show that it can be a valuable tool for environmental time series analysis. Our aim is to contribute to the community of TSC, here focusing on the CCS, geo-science and oceanography applications.

A recent technique for gas seep detection is based on relatively simple threshold techniques of the difference between time lags in time series [39], and we are certain that our study can be a contribution to increase detectability and thus optimization of monitoring programs in future CCS projects.

We present a solution to the binary classification problem of detecting CO₂ seeps in the marine environment using the Scottish Goldeneye area as the case study.

Classifying time series requires data for all the classes, i.e., time series of the variables in question for no-leak and leak conditions. The no-leak situation represents natural environmental statistics, i.e., the environmental baseline, and should be based on in situ measurements, preferably supplemented with model simulations [40]. Time series for the seep situations must rely on process modelling, simulating the different processes involved during a seep [41–44], preferably supported by in situ and laboratory experiments [45]. The use of modeling data in this case is a necessity since the data corresponding to the leak scenarios are difficult, expensive, and, in some cases, impossible to obtain.

Moreover, the trained deep neural network can be used in a transfer learning setting [46], i.e., we can pre-train a neural network on the model data, fix the parameter weights in the first few layers, and then train the network with the limited in situ measurement data as input. The conjecture is that the first few layers adequately represent the core feature characteristics of the time series, and thus reduce the need for in situ data.

Here we use model data from the Goldeneye area, in the Scottish sector of the North Sea, obtained through the STEMM-CCS project (<http://www.stemm-ccs.eu>). The simulations are performed with the

Finite-Volume Coastal Ocean Model (FVCOM) [47] coupled with European Regional Seas Ecosystem Model (ERSEM) [48]. Different scenarios have been created in a statistically sound manner to train the BCNN. Deep learning can be used as surrogate models to extend the results from PDE simulators; see e.g., Hähnel et al. [49] and Ruthotto and Haber [50]. Here instead we simulate data with a computational fluid dynamics model, and use neural networks to estimate model parameters.

This manuscript is outlined in the following manner: In Section 2 we present the underlying framework for Monte Carlo dropout (MC dropout) and BCNN in a binary TSC context. We present a general deep-learning framework, Bayesian Neural Networks, how the stochastic regularization technique MC dropout can produce predictive distributions, Bayesian decision theory and presents an algorithm for decision support in environmental monitoring under uncertainty. In Section 3, we apply the MC dropout for the case study at the Goldeneye area. Here we describe the data and how it is pre-processed, present the model, architecture and hyper-parameter settings. We use the output of the classifier in a Bayesian decision rule setting with varying cost function and demonstrates our proposed algorithm. Section 4 summarize our findings, compare our approach with relevant literature, discuss strengths and weaknesses and proposes potential extensions and further work.

2. Methods

MC dropout and BCNN was introduced by Gal and Gharmani in [28,51,52] and we follow their notation in this section.

2.1. Problem Formulation

Let $\mathbf{x} \in \mathbb{R}^m$ be a time series of m observations. We assume that any time series $\mathbf{x} \in \mathbf{X}$, where \mathbf{X} is a large set of N time series of the length m , can be assigned to either the no-leak or the leak class. In what follows, we label \mathbf{x} with the vector $\mathbf{c}_0 := (1, 0)$ if it corresponds to the no-leak class, and with $\mathbf{c}_1 := (0, 1)$ for the leak class. The labeled time series $\{\mathbf{x}, \mathbf{y}\}$, $\mathbf{y} \in \{\mathbf{c}_0, \mathbf{c}_1\}$ is referred to as an instance, and the ordered set of instances (\mathbf{X}, \mathbf{Y}) as a data set. Time series could be pre-processed measurements of, e.g., pH or tracer concentrations. The task of binary TSC is to design a classifier that is a function that maps the time series \mathbf{x} to a probability of a class $p(\mathbf{y} = \mathbf{c}_i)$, $i = 0, 1$ based on the training data (\mathbf{X}, \mathbf{Y}) . As $p(\mathbf{y} = \mathbf{c}_0) = 1 - p(\mathbf{y} = \mathbf{c}_1)$, we simply write $p(\mathbf{y})$ instead of $p(\mathbf{y} = \mathbf{c}_1)$ further on if no clarification is needed. The classifier can be approximated by a universal approximation such as an artificial neural network. A traditional neural network has the disadvantage of having no knowledge of the error of the classifier approximation. This shortcoming can be resolved by using a Bayesian framework which allows for distributions over the model weights. Sampling randomly from the distribution of network weights and predicting with each sampled weight results in a posterior predictive distribution of the class $p(\mathbf{y})$.

2.2. Artificial Neural Network

Here we use a single hidden layer model for the convenience of notation. The framework can be expanded to arbitrary number of hidden layers. A single hidden layered neural network is defined as

$$\hat{\mathbf{y}} = \mathbf{f}^{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2}(\mathbf{x}) := S_2(S_1(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2).$$

Here $\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2) \in \mathbb{R}^2$, S_1 and S_2 are activation functions and $\omega = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2\}$ are the parameters of network model. In particular, $\mathbf{W}_1 \in \mathbb{R}^{m \times k}$, $\mathbf{W}_2 \in \mathbb{R}^{k \times 2}$, and $\mathbf{b}_1 \in \mathbb{R}^k$, and $\mathbf{b}_2 \in \mathbb{R}^2$. In this study we use convolutions to transform the input to the hidden layer. This means that we restrict the structure of the weight matrix \mathbf{W}_1 to be a convolution and instead of learning the best fit over all possible weight matrices learn the best fit over all weight matrices that are convolutions. Using convolutions in this transformation is what is referred to as a convolutional layer.

To obtain the probability of \mathbf{x} being classified with a label \mathbf{y} , we use a SoftMax activation function, i.e.,

$$S_2(\mathbf{z}) = \frac{\exp(\mathbf{z})}{\exp(z_1) + \exp(z_2)}.$$

In classification the cross entropy loss function is the natural choice, and minimizing the cross entropy is equivalent to minimizing the negative log likelihood

$$E^\omega(\mathbf{X}, \mathbf{Y}) = - \sum_{n=1}^N \mathbf{y}_n \log(\hat{\mathbf{y}}_n) = - \log p(\mathbf{Y} | \mathbf{f}^\omega(\mathbf{X})).$$

To avoid the model overfitting on the training set, which typically results in a failure to generalize the model, regularization is often added. One of the choices is L_2 regularization, which penalize the model parameters with squared L_2 norm and gives the following objective function

$$C(\omega) = - \log p(\mathbf{Y} | \mathbf{f}^\omega(\mathbf{X})) + \lambda_1 \|\mathbf{W}_1\|_2^2 + \lambda_2 \|\mathbf{W}_2\|_2^2 + \lambda_3 \|\mathbf{b}_1\|_2^2 + \lambda_4 \|\mathbf{b}_2\|_2^2, \quad (1)$$

where $\lambda_i > 0, i = 1, \dots, 4$ are regularization parameters. The importance of L_2 regularization will become evident in Section 2.4. Minimizing the objective Equation (1) with respect to the parameters ω , through the techniques of back-propagation and stochastic gradient descent is the core task in machine learning. This process will give a point estimate for each model parameter. In many situations, it is important to quantify the uncertainty of the prediction outcome. In this case a Bayesian take on the problem is suitable.

2.3. Bayesian Neural Networks and Bayesian Parameter Estimation

In a neural network for binary TSC we want to estimate parameters or weights ω that best determine which class the time series belongs to. For a Bayesian Neural Network, any knowledge we have on the weights before training are referred to as the prior and denoted $p(\omega)$. The prior can be updated after observing a new time series (\mathbf{X}, \mathbf{Y}) , to reflect the most likely values of ω

$$p(\omega | \mathbf{X}, \mathbf{Y}) = \frac{p(\omega)p(\mathbf{Y} | \mathbf{X}, \omega)}{p(\mathbf{Y} | \mathbf{X})}.$$

Here, $p(\omega | \mathbf{X}, \mathbf{Y})$ is called the posterior distribution of ω , while $p(\mathbf{Y} | \mathbf{X}, \omega)$ is referred to as the model (here the neural network architecture) or likelihood function. The marginal likelihood $p(\mathbf{Y} | \mathbf{X})$ is defined as the integral

$$p(\mathbf{Y} | \mathbf{X}) = \int p(\mathbf{Y} | \mathbf{X}, \omega) p(\omega) d\omega.$$

The posterior predictive distribution is defined such that,

$$p(\mathbf{y}^* | \mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}^* | \mathbf{x}^*, \omega) p(\omega | \mathbf{X}, \mathbf{Y}) d\omega, \quad (2)$$

where \mathbf{x}^* represent a new observation with unknown target value \mathbf{y}^* . By varying ω , Equation (2), can be viewed as an ensemble of models weighted by $p(\omega | \mathbf{X}, \mathbf{Y})$. It is difficult and sometimes impossible to solve Equation (2) analytically, thus we often resort to Monte Carlo (MC) sampling. If we have the distribution over the weights in the neural network we can simply sample from the distribution of these to get the posterior predictive distribution, i.e., estimate of the uncertainty of the time series class predicted by the neural network. If the Monte Carlo sampling becomes too computational costly, we can turn to other approximation methods such as variational inference or MC dropout.

2.4. Monte Carlo Dropout

Dropout [53] is a stochastic regularization technique where during training, noise in the model is introduced by randomly setting a proportion of the nodes in the model to zero in each batch. Which nodes that are set to zero is determined by a Bernoulli distribution. During prediction dropout is turned off, resulting in a point estimate of class probabilities. MC dropout is basically the same; however, during prediction dropout is still turned on, randomly shutting a proportion of the nodes off. In this way, dropout generates a distribution over the model parameters by repeating the nodes sampling several times and predicting for each configuration. The process is similar to a bootstrap procedure [54]. MC dropout thus produces a posterior predictive distribution for the class probabilities. The process of dropping out units in the feature space (i.e., dropping out nodes) can be translated to the parameters space.

We denote the dropped out weights $\hat{\omega} = \{\widehat{\mathbf{W}}_1, \widehat{\mathbf{W}}_2, \mathbf{b}_1, \mathbf{b}_2\}$ which allow for a convenient representation of the MC dropout objective function. Here we also use data sub-sampling (mini-batches) with random index set S of size M and

$$\hat{\omega}_i = \{\widehat{\mathbf{W}}_1^i, \widehat{\mathbf{W}}_2^i, \mathbf{b}_1, \mathbf{b}_2\} = \{\text{diag}(\hat{\epsilon}_1^i) \mathbf{W}_1, \text{diag}(\hat{\epsilon}_2^i) \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2\},$$

with $\hat{\epsilon}_1^i \sim p(\hat{\epsilon}_1^i)$ and $\hat{\epsilon}_2^i \sim p(\hat{\epsilon}_2^i)$, for $1 \leq i \leq N$ where $p(\epsilon_1)$ and $p(\epsilon_2)$ is Bernoulli distributions with probabilities p_1 and p_2 respectively. Thus, we define the objective function over the mini-batch sets such that

$$\begin{aligned} \widehat{\mathcal{L}}(\omega) = & -\frac{1}{M} \sum_{i \in S} \log p(\mathbf{Y}_i | \mathbf{f}^{\hat{\omega}_i}(\mathbf{X}_i)) \\ & + \lambda_1 \|\mathbf{W}_1\|_2^2 + \lambda_2 \|\mathbf{W}_2\|_2^2 + \lambda_3 \|\mathbf{b}_1\|_2^2 + \lambda_4 \|\mathbf{b}_2\|_2^2. \end{aligned} \quad (3)$$

Trough back-propagation and stochastic gradient decent we find optimal parameters for the model \mathbf{f}^ω as

$$\arg \min_{\omega} \widehat{\mathcal{L}}(\omega).$$

We must integrate over the weights to get the posterior predictive distribution, as shown in Equation (2). With weights optimized we can use the model to predict new classes given some new input \mathbf{x}^* . To approximate the posterior predictive distribution we use a MC estimator to integrate over the weights. The weights $\hat{\omega}_i \sim p(\omega | \mathbf{X}, \mathbf{Y})$ are generated from the posterior distribution such that,

$$p(\mathbf{y}^* | \mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}^* | \mathbf{x}^*, \omega) p(\omega | \mathbf{X}, \mathbf{Y}) d\omega \approx \frac{1}{T} \sum_{t=1}^T \mathbf{f}^{\hat{\omega}_t}(\mathbf{x}^*) \xrightarrow{T \rightarrow \infty} \mathbb{E}[\mathbf{y}^*],$$

which is an unbiased estimator and what we referred to as MC dropout.

In [51] Gal showed that variational inference could be approximated by MC dropout. This is the case if the derivatives of the two objective functions corresponding variational inference and MC dropout, with respect to the model parameters, are equal. This is true if the so-called KL condition is satisfied. In the same paper, Gal showed that Gaussian priors over the model parameters satisfy this KL condition for large enough hidden units. So, with Gaussian priors, MC dropout approximates variational inference. While there could be some other priors that satisfy the KL condition, MC dropout and variational inference are not generally equivalent. For this reason we use the Gaussian priors, which are well known to be equivalent to L_2 regularization. We refer the reader to Gal and Ghahramani [28,51] for details.

2.5. Uncertainty Estimation in MC Dropout

In regression task Gal and Ghahramani [28] derived an analytical estimate of the predictive mean and variance. For classification, they only presented an analytical estimate for the first

order moment. That means other measures of the uncertainty must be obtained in a classification setting. One approach would be to model the log-probabilities instead and calculate the mean and standard error of the log-probability. In order to keep it simple, we have adopted the strategy by Leibig et al. [31]. They solved a binary classification problem on images with MC dropout (actually a Bayesian Convolutional Neural Network) and used the empirical standard deviation of the positive class as an estimate of the uncertainty in the prediction. We adopt this intuitive estimate of uncertainty in our presentation of results. The empirical mean of the predicted leak is defined as

$$\hat{\mu}_{Leak} = \frac{1}{T} \sum_{t=1}^T p(\mathbf{y}^* = \mathbf{c}_1 | \mathbf{x}^*, \hat{\omega}_t),$$

and empirical standard deviation as

$$\hat{\sigma}_{Leak} = \sqrt{\frac{1}{T} \sum_{t=1}^T [p(\mathbf{y}^* = \mathbf{c}_1 | \mathbf{x}^*, \hat{\omega}_t) - \hat{\mu}_{Leak}]^2},$$

where T is number of forward passes.

2.6. Bayesian Decision Making

We want to use the information gained about the uncertainty of the TSC to decide about whether or not consider the classification as a leak or no-leak situation. Bayesian decision theory [55] can be used to make an optimal decision based on cost related to the different choices one could make. We describe a Bayesian decision rule with loss functions and present an algorithm for probabilistic decision support in Section 2.7. This algorithm is applied and showcased through an example in Section 3.7.

A loss function in terms of Bayesian decision-making states how costly an action or decision is. In a binary classification setting we have two classes $c : \{c_1, c_2\}$ and a possible actions $\{\alpha_1, \dots, \alpha_a\}$. The loss function $\lambda(\alpha_i | c_j)$ is the cost associated with taking action α_i if the class is c_j . By considering the loss associated with a decision we can define the expected loss or conditional risk

$$R(\alpha_i | \mathbf{x}) = \sum_{j=1}^2 \lambda(\alpha_i | c_j) P(c_j | \mathbf{x}). \quad (4)$$

Here, $P(c_j | \mathbf{x})$ is the posterior predictive probability for a given class c_j , given a time series \mathbf{x} . Remember, the predictive posterior distribution is estimated through MC estimation from the BCNN for each time series. Given a time series \mathbf{x} it is possible to minimize the expected loss by choosing the action that minimizes the conditional risk. A decision rule R is a function that takes the input time series to a space of possible actions $R: \mathbb{R}^d \rightarrow \{\alpha_1, \dots, \alpha_a\}$. The expected loss of a decision rule can be stated as

$$L = \int R(\alpha(\mathbf{x}) | \mathbf{x}) p(\mathbf{x}) d\mathbf{y}.$$

The aim is to use the rule $\alpha(\cdot)$ that minimizes $R(\alpha(\mathbf{y}) | \mathbf{y})$ for all \mathbf{y} . Minimize the conditional risk based on the actions is in fact the optimal decision given our information

$$\begin{aligned} \alpha^* &= \arg \min_{\alpha_i} R(\alpha_i | \mathbf{x}) \\ &= \arg \min_{\alpha_i} \sum_{j=1}^c \lambda(\alpha_i | c_j) P(c_j | \mathbf{x}). \end{aligned}$$

In the case of only two possible actions and classes, the conditional risk can be expressed as

$$\begin{aligned} R(a_1|x) &= \lambda_{11}P(c_1|x) + \lambda_{12}P(c_2|x), \\ R(a_2|x) &= \lambda_{21}P(c_1|x) + \lambda_{22}P(c_2|x). \end{aligned}$$

The decision rule then becomes to choose the actions which give the smallest overall risk. We choose action a_1 if

$$R(a_1|x) < R(a_2|x).$$

From Equation (4) we have that the risk can be expressed in terms of the loss function and the posterior distribution. In the two class and action case, we decide action a_1 if

$$(\lambda_{21} - \lambda_{11})P(c_1|x) > (\lambda_{12} - \lambda_{22})P(c_2|x).$$

We want to use the concepts outlined above for the classified time series and their posterior predictive distribution. This results in a novel approach for decision support in environmental monitoring under uncertainty.

2.7. Decision Support in Environmental Monitoring under Uncertainty

In context of decision-making, we need to define a posterior summary function Γ that summarizes the posterior distribution. Examples of posterior summary functions are the expectation and the maximum a posterior (MAP) of the predictive posterior distribution generated from the BCNN. The posterior predicted expectation can be approximated as the average of the realizations

$$P(c_j|x) = \Gamma(\{P_t(c_j|x)\}_{1 \leq t \leq T}) \approx \frac{1}{T} \sum_{t=1}^T P_t(c_j|x). \quad (5)$$

Alternatively, we can use the mode or maximum a posterior probability (MAP) of the sample distribution, i.e., the probability that most often occur in the predictive posterior distribution. The posterior predictive distribution do not have a uniform discretization, and each sample may be unique. A function \mathcal{H} takes $P_t(c_j|x)$ as input and maps the realizations t into K bins with consecutive, non-overlapping intervals such that $\mathcal{H}(P_t(c_j|x)) = P_k(c_j|x)$, where $k = 1, \dots, K$. An estimate of the predictive posterior distribution through the mode/MAP is then to find the bin with most counts and its associated probability

$$P(c_j|x) = \Gamma(\{P_t(c_j|x)\}_{1 \leq t \leq T}) \approx \mathcal{H}(P_t(c_j|x)) \approx \arg \max_{k \in \{1, \dots, K\}} P_k(c_j|x) / K \quad (6)$$

With Equations (5) and (6) in mind, we present a three-step procedure for probabilistic decision support in environmental monitoring which is presented in pseudo-code in Algorithm 1.

Step 1 in Algorithm 1 is costly; however, this is not the case of step 2 and 3. The majority of the time will be devoted to optimizing the weights of the BCNN. When step 1 first has been performed, that is training of the BCNN, step 2 and step 3 can be conducted independent of step 1.

Algorithm 1 Algorithm for decision support in environmental monitoring under uncertainty**Input:**

- Training set \mathbf{X}, \mathbf{y}
- Unlabeled time series \mathbf{x}^*
- Number of realizations in posterior sampling T
- Number of classes C
- CCN model \mathbf{f}^ω with weights ω
- Posterior summary function $\Gamma(\{P_t(c_j|\mathbf{x})\}_{1 \leq t \leq T})$
- $\lambda(\alpha_i|c_j)$ cost associated with taking action α_i if the class is c_j

1. **Optimize CNN model weights with MC dropout algorithm**
 $p(\omega|\mathbf{X}, \mathbf{y}) \leftarrow$ Optimize BCNN model
 2. **Generate posterior predictive distribution from optimized BCNN**
 $\hat{\omega}_t \sim p(\omega|\mathbf{X}, \mathbf{y}) \leftarrow$ Simulate T samples from the posterior distribution of the weights
 $p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) \approx \mathbf{f}^{\hat{\omega}_t}(\mathbf{x}^*) \in \mathbb{R}^{C \times T} \leftarrow$ Estimate posterior predictive distribution for all classes
 $P_t(c_j|\mathbf{x}^*) = (\mathbf{f}^{\hat{\omega}_t}(\mathbf{x}^*))_j \in \mathbb{R}^T \leftarrow$ Extract the posterior distribution for class c_j with T samples
 $P(c_j|\mathbf{x}^*) = \Gamma(\{P_t(c_j|\mathbf{x})\}_{1 \leq t \leq T}) \leftarrow$ Approximate $P(c_j|\mathbf{x}^*)$ with e.g., (5) or (6)
 3. **Make optimal decision based on posterior predictive distribution**
 $\alpha^* = \arg \min_{\alpha_i} \sum_{j=1}^C \lambda(\alpha_i|c_j) P(c_j|\mathbf{x}^*) \leftarrow$ Minimize the conditional risk.
- return** $\alpha^* \leftarrow$ Optimal decision α_i based on $P(c_j|\mathbf{x}^*)$ and cost function $\lambda(\alpha_i|c_j)$

3. Case Study—Goldeneye CCS Site**3.1. Data**

The data used in this study has been produced with FVCOM coupled with a biochemical tracer model, ERSEM via the framework for aquatic biogeochemical models coupler [48]. This framework provides spatio-temporal time series (4D) for both carbon chemistry and biological processes, and can model the natural variability of CO₂ transportation in the oceans. Furthermore, it is possible to produce artificial leaks on the seafloor that blend with the natural variation, to produce realistic simulations of CO₂-leaks. Cazenave et al. [56] produced a data set to present an approach to design marine monitoring networks for CO₂ storage, where they used a weighted greedy algorithm to identify, based on a limited number of sample stations, spots that give best possible coverage. Here we adopt the CO₂ concentration data from Cazenave et al. simulations, after some preprocessing steps, as input for the probabilistic deep-learning framework. The data from Cazenave et al. consist of four different simulations, one without any leakage, i.e., only the natural variability, and three with leak at the center of the domain with three different release rates. These different simulations are referred to as scenarios, and are labeled 0T (or no-leak), 30T, 300T and 3000T. The labeling are based on the amount of tons CO₂ that is released per day in each simulation. We refer the reader to the paper [56] for more details about the simulations, setup and the different scenarios in general. Figure 1 shows the depth at the Goldeneye area where the leak simulations was conducted.

3.1.1. Description of Data Set

The simulation output consists of a 4D dataset, three spatial dimensions; latitude, longitude and depth as well as a fourth time dimension. There are in total 1736 nodes and 24 layer that make up an irregular triangulation in the latitude-longitude dimension and the resolution is refined near the leak. In the depth dimension sigma layers are used that is the resolution varies with the bathymetry. There are $1736 \times 24 = 41,664$ (nodes and layers) time series for each simulation and the time series has 1345 time steps. The time between each time step is 15 min, which means the simulations last for approximately 14 days. The data has been split in two with respect to time, where the layer closest to

the seafloor in the second half of the data set is used for testing and the first half of the time series is used for training and validation. The 4 simulations we received resulted in a total of 166,656 time series of which 70% of the data in the first split have been used for training and 30% validation. After labeling the data (see Section 3.1.2), the training and validation data consisted of 24.2% and 75.8% time series labeled as leak and no-leak respectively. The test data consisted of 36.2% time series labeled as leak and 69.8% time series as no-leak. See Table 1.

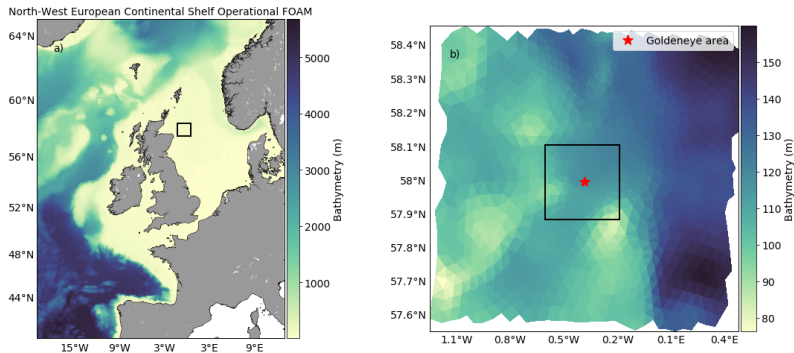


Figure 1. (a) FVCOM domain used in which resolution varies from 15 km at the open boundaries to 0.5 km at the release site. The black box indicates the extents of the grid shown in (b). (b) The nested domain with resolution from 0.5 km at the boundary to 3 m at the release site (red star). The black box in (b) indicates the extent of the Goldeneye complex [56].

Table 1. Overview of the train, validation and test data.

	Leak	No-Leak	# Time Series	Start/End
Training Data	75.8%	24.2%	116,659	Start
Validation Data	75.8%	24.2%	49,997	Start
Test Data	69.8%	36.2%	6944	End

3.1.2. Preprocessing of Data

There have been five main steps of preprocessing before the data is fed to the model: Removal of time steps, splitting of the time series into two, labeling of data, time lag differencing and standardization of the data. First of all we remove the first 245 time steps of the original data due to a spin-up period of the hydrodynamic simulations, such that the data to be used further have a time dimension of 1100. Secondly, we split the data in two, using the first 550 time steps as training and validation and the last 550 time steps as test data. This is done so the test data becomes less correlated with the training and validation data sets. One of the simulations are without any leakage, and by subtracting the relevant values from the simulation with leak from the one without gives the *true footprint* of the leakage. Based on the true footprint a threshold can be chosen to label time series as either leak or no-leak. The recommended uncertainty measure from [57] on dissolved inorganic carbon is $1 \mu\text{mol/kg}$ and corresponds to approximately 1.0236 m mol/m^3 (which is the output of the hydrodynamic simulations) with a density of seawater of 1023.6 kg/m^3 . The density will change with depth and location. Here we use a threshold of 0.01 mmol/m^3 such that if the true footprint at some point has been above this value the corresponding time series will be labeled as a leak. The two last steps of the preprocessing is to apply the difference transform and standardize the data. Applying a difference transform can help stabilize the mean of the time series by removing changes in the level

of a time series and so eliminating and reducing trend and seasonality. Instead of assessing the time series itself changes in the signal is analyzed (an approximation of the derivative). Blackford et al. studied changes in the pH to determine if a leak was present in [39]. It makes sense to apply the difference transform in an anomaly detection perspective, where abnormal changes and patterns in the time series can be a good indicator if a leakage is present or not. Standardization of the data should improve the conditioning of the optimization problem, and speed up the training process.

3.2. Model for TSC: Bayesian Convolutional Neural Networks

We use a traditional 1D Convolutional Neural Network model to classify time series. The model has 4 convolutional layers with ReLu activation functions [58], each followed by a MC dropout and MaxPooling layer, where MaxPooling is a down-sampling technique in convolutional neural networks. The first and last convolutional layers have 128 filters, while the middle layers both have filter size of 256. The kernel size of the convolutional layers is $\{7, 5, 3, 2\}$ from first to last and we apply a stride of 1. We use the same dropout rate of 50% for all MC dropout layers and all MaxPooling layers have a pool size of 2. The dropout rate of 50% gives the largest variance over the model weights. Interpreting dropout as an ensemble of networks, a dropout rate of 50% produces most possible combinations, and thus the largest variability over weights and posterior predictive distribution. According to Baldi et al. [59], the 50% rate results in the strongest regularization effect, but has the disadvantage of potentially worse convergence than other dropout rates. There may exist dropout rates that gives better accuracy. However, to maximize the variance of the predictive posterior distribution, 50% rate is favorable. The last part of the model is a flattening layer followed by a dense layer with two nodes and SoftMax activation function, resulting in two SoftMax probabilities, one for each class. An illustration of the model is presented in Figure 2.

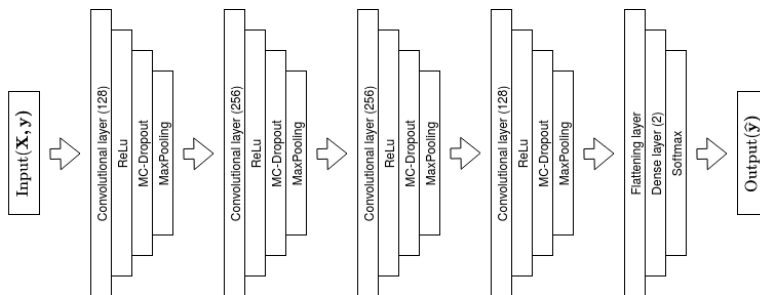


Figure 2. Illustration of the Bayesian Convolutional Neural Network Model used.

Mathematically we can express the model in terms of the weights, bias and activation functions

$$\hat{\mathbf{y}} = \sigma_S(\sigma_R(\sigma_R(\sigma_R(\mathbf{x}\hat{\mathbf{W}}_1 + \mathbf{b})\hat{\mathbf{W}}_2)\hat{\mathbf{W}}_3)\hat{\mathbf{W}}_4)\mathbf{W}_5.$$

Here the subscript S and R refer to a SoftMax and ReLu activation functions, respectively. The $\hat{\mathbf{W}}_l$ notation, indicates that nodes are dropped out with a Bernoulli distribution, and that there are in total four layers $l = \{1, 2, 3, 4, 5\}$. It is only the first four layers that include MC dropout regularization. The model has 435842 trainable parameters and for optimization of the we use the well-known ADAM [60] algorithm. The goal is to find the parameters $\phi = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3, \mathbf{W}_4, \mathbf{W}_5, \mathbf{b}\}$ such that the $\hat{\mathcal{L}}_{dropout}(\phi)$ is minimized; see Equation (3). A early stopping regime is also introduced to avoid overfitting and the validation loss is monitored and the optimization stopped after 50 epochs if no improvement is observed. We used a batch size of 256 and monitored both the accuracy and the loss. An illustration of the convergence of the training and validation of the model is showed in Figure 3.

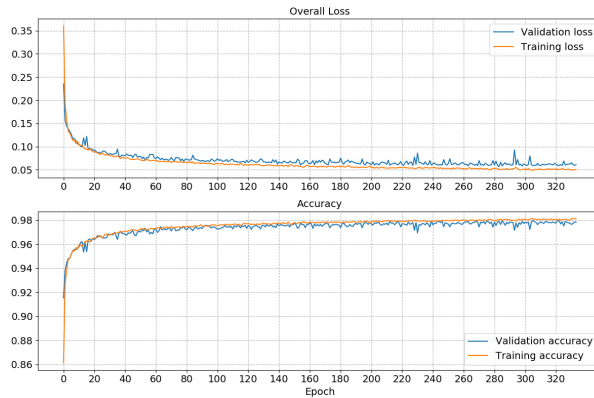


Figure 3. Convergence of the BCNN. In total it ran for 334 epochs, approximately two hours of train/validation time with a NVIDIA Titan V GPU.

3.3. Performance of the Classifier

Figure 4 shows the proportion of true positive vs. true negatives and the Area Under the Curve (AUC). Higher the AUC indicates that the model is better at distinguishing between leak and no leak. Given the costs associated with true positives/negatives and false negatives/positives these characteristics will be useful for designing monitoring program and decision-making in mobilizing the leak confirmation phase.

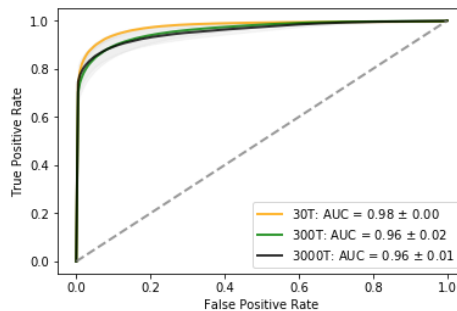


Figure 4. ROC-curves for the three leak scenarios. The 0T scenario is not included since it in all cases will be classified as no-leak, i.e., there are no false positives.

In Figure 5, we show two histograms of the prediction probability and standard deviation split by the different scenarios. If we concentrate on the edges on the prediction probability histogram we see that the majority of the time series is classified as no-leak or leak and falls within the first and last bins, indicating good classification capabilities. The 0T scenario is not represented on the right-hand side of the histogram which is in line with our expectations as the classifier should not predict leaks when there is none. Smaller leaks have also fewer time series that are classified as leaks, which is intuitively since leaks with larger flux have a more distinct footprint that can be transported further. Assessing the histogram of the standard deviation, it is observed that smaller leaks scenarios have more spread than larger ones. A larger leak has a stronger signal and thus is easier to classify.

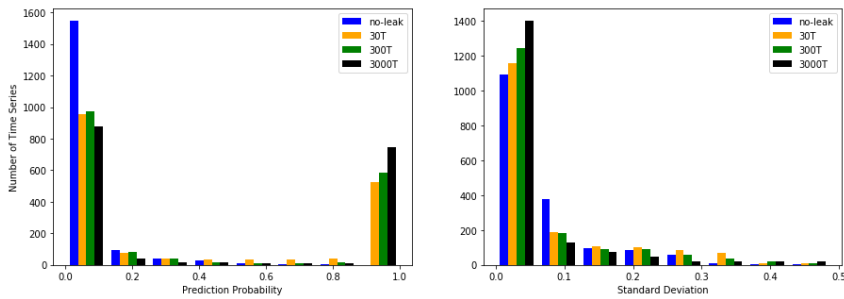


Figure 5. (Left Panel) Prediction probability for each scenario plotted as a histogram. (Right Panel) Standard deviation for each scenario plotted as a histogram. The central observation is that most of the time series is classified as either leak = 0 or No-leak = 1. Minor leaks have a smaller proportion of the time series that are classified as leaks than larger ones and smaller leaks are associated with a higher degree of uncertainty than the larger ones.

In Figure 6 the rolling mean and standard deviations with respect to the different scenarios is plotted against the distance from the leak location. We have used a quite high window size of 50 to catch and visualize trends in the data. 30T, 300T and 3000T scenario have a high confident in its prediction to about 0.1, 0.3 and 1–2 km, respectively. After this the classifier becomes less confident that there is a leakage present. For the 0T scenario a corresponding decrease in confident is observed, until it reaches its peak around 4 km. The most uncertain area of the 0T scenario coincides with the most uncertain area of the leak scenarios. That is the region from approximately 1 to 10 km from the leakage.

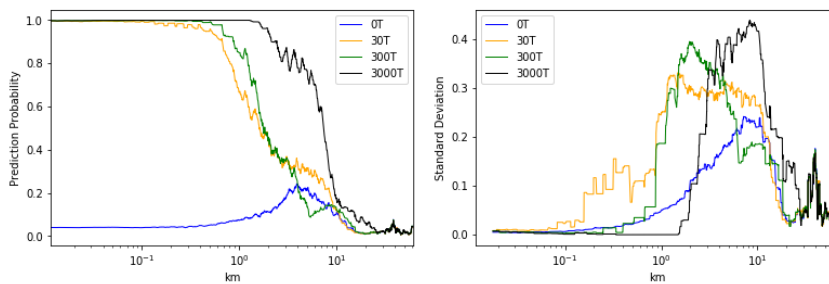


Figure 6. (Right panel) A moving mean of the prediction probability vs. the distance from the leakage. (Left panel) A moving standard deviation vs. the distance from the leakage. For the moving statistics, all points are evenly weighted.

Calculating the area under the graph for each of the scenarios in Figure 6 gives us information on the detectability of each scenario and what scenario that are associated with highest uncertainty. The 0T scenario have the lowest area under the graph for both the prediction probability and the standard deviation, meaning that on average this is the scenario associated with highest predictive power but also the lowest uncertainty (For the 0T scenario the target is a prediction probability of 0 and not 1 as for the other scenarios). With increasing flux we observe increased area under the graph, indicating that larger flux is related to detection further away from the source. It is also observed a decrease in area under the graph for the standard deviation, indicating that smaller leaks are associated with higher degree of uncertainty. We refer to both Table 2 and Figure 6.

Table 2. Approximation of area under the graph for the different scenarios for both prediction and standard deviation.

Scenario	Prediction Probability	Standard Deviation
0T	90.15	115.48
30T	624.25	183.54
300T	633.58	161.82
3000T	773.77	153.94

Figure 7 shows a 2D histogram of the entire test data set, i.e., all scenarios combined, where the prediction probability is plotted against the standard deviation. Due to high density around 0 and 1 the color that represent the density has log-scale. As in showed in Figure 5 it is observed that the majority of the time series are close to 0 and 1, i.e., the model classifies with high confident either leak or no-leak. There are however time series that are difficult to distinguish, which have both high standard deviation and indefinite prediction probability. Making a decision under such conditions is difficult. The framework for how to decide what action to take under some cost is discussed and presented in Section 3.7.

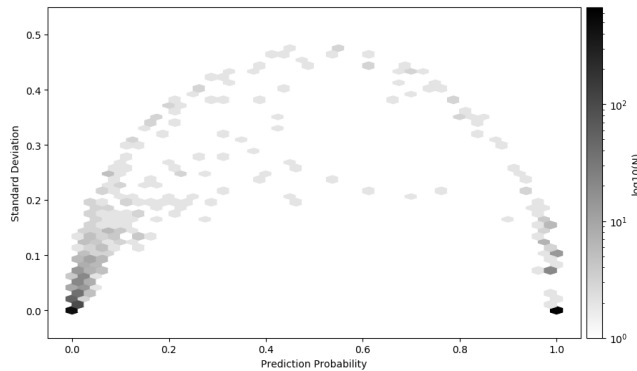


Figure 7. 2D histogram of the mean prediction and standard deviation of all the time series in the test data set. The majority of the time series are predicted near 0 or 1 with low standard deviation. The color pallet have log-scale to visualize the time series that are classified with high standard deviation and low distinction in the prediction probability.

3.4. Approximated Predictive Mean and Uncertainty

In Figure 8 we see that the accuracy around the leakage is high. We also observe areas with quite high uncertainty. Due to the nature of the deep-learning methods, it is difficult to get insight in why some regions are more uncertain than other. We can only expect the classifier to detect leaks if traces of the leak have reached that particular point. There may be several reasons why we see this behavior: The training and validation data may be too different from the test data. Time series data from the two classes may be too alike, making it very difficult to distinguish the classes, or the natural variability and a leak behave in some regions similar. The classifier will in these cases have a hard task, and in some cases misclassify or be uncertain about the prediction.

Figure 9 shows kernel density estimates (KDE) of the 200 forward MC realizations for the three different locations in Figure 8. The yellow KDE is from the area south, in a region with high uncertainty. The distribution is almost uniform, making it very difficult to say if the time series arise from a leak situation. The blue KDE predicts that the time series arise from a leakage situation, with quite high confidence. This is reasonable, as the measurement is quite close to the leak locations. The cyan KDE

shows relatively little spread, with a clear indication that this arise from a no-leak situation. Because of the distance from the leak location, this is reasonable, as traces from the leak might not reached this region of the area. One of the key concepts here has been to show how to model uncertainty in time series with modern deep-learning techniques. We therefore did not put a tremendous effort into optimization of hyper-parameters of the BCNN to improve the accuracy and decrease the uncertainty.

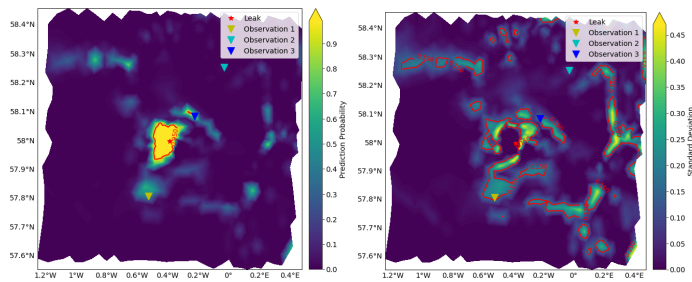


Figure 8. Prediction from the BCNN on the 3000T test data. **(Left panel)** The plot shows the predictive mean leak probability of the 1736 nodes with 200 forward MC realization on each instance. The red line shows where the predictive mean leak probability is above a value of 0.95. **(Right panel)** The plot shows the uncertainty in the prediction. Red line shows where the standard deviation is above 0.15 indicating areas where the prediction is uncertain. See Figure 9 for more details about the uncertainty in observation locations 1, 2, and 3.

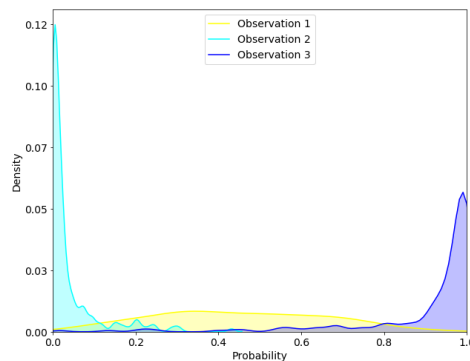


Figure 9. Kernel density estimation (KDE) of the three observations in Figure 8. We have used the seaborn visualization library, i.e., Gaussian kernel with Scott method for estimation the kernel bandwidth. The KDE smoothens the empirical distribution, thus exceeding the estimate beyond the possible range of $[0, 1]$. We thus limit the plot to be within the bounds, which means that this KDE does not summarize to 1.

3.5. Detectable Area vs. Detection Probability

In Figure 10, we have plotted the area covered against the detection probability. First we observe that the detected area is greater with larger leakages. Secondly, the 30T and 300T scenario have small difference in percentage of area detected up to approximately 80% detection probability. Thirdly, if we demand high prediction probability the different leak scenarios can detect an area around 1–10%, 0.1–1% and 0.1–0.01% for the 3000T, 300T, and 30T leak scenario, respectively. This gives us insight on the size of the footprint we can detect and thus some insight in how to optimize sensor layout.

Simulated leaks have been used to predict footprints subsequently being used to optimize sensor layout [61–65]. They used a threshold that was based on the Cseep method [66] that is dependent on measurements of different biochemical tracers. We argue that using deep-learning methods and BCNN can lower this threshold, increasing the detectable footprints and thus the decreasing the number of sensors needed to monitor a specific area.

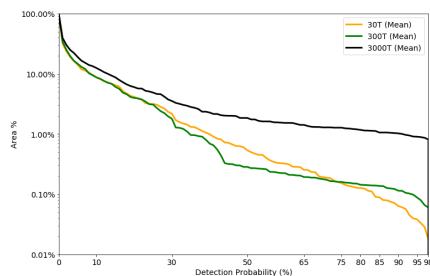


Figure 10. Mean detection probability plotted vs. the total area covered at specific threshold.

3.6. Sensitivity Analysis

We have validated the model in the bottom layer only, because this is where the sensors can be placed. We have additionally made two sanity checks on the generalization properties of the approach. The first is related to how sensitive the method is if we train only on the simulations with 0T, 30T and 3000T and test this new trained model on the test data set. The second address the sensitivity if we add some stochastic Gaussian noise to the test data.

3.6.1. Reducing the Training Data Set

The model was optimized similar to as described in Section 3.2, the only difference is that the 300T simulation scenario is removed from the training data set. The ROC for the case where the 300T scenario is removed from the training data set is presented in the right panel of Figure 11, with associated AUC values. We observe that the AUC value is relatively high in the case with 300T, despite no training on leaks with such size. The overall AUC is high in all cases, but with an increased uncertainty compared to AUC values in Figure 4. In general the AUC values are relatively large compared to training with all simulated scenarios. This suggest that the model generalizes well.

3.6.2. Adding Gaussian Noise the Test Data Set

We briefly assess the classifiers sensitivity by adding Gaussian Noise, $\mathcal{N}(0, \sigma)$, to the test data set, before we predict by using the same trained model as above. The noise was added to the pre-processed time series of the test data set. We tested two cases, where we chose $\sigma = 0.01$ and $\sigma = 0.1$.

The general observation for a relatively low level of noise ($\sigma = 0.01$), is that time series with true label no-leak had a higher uncertainty related to its prediction and the predicted value of it being labeled as leak increased. That is the AUC has dropped from values near 1, see Figure 4, to values around 0.8, see Figure 11. The prediction of true leaks were improved, however at the cost of more false positives. With a noise level of 0.1, the majority of the time series were classified as leaks. This tells us that this classifier is sensitive to small and rapid changes in the data, for data where the true label is no-leak, which can lead us to two potential conclusions. Either rapid and relatively small changes is a potential indicator of a leakage, or the classification is just comprised due to the additional noise. There are studies using rapid changes as indicators of leaks, as described by Blackford et al. in [39]. This also highlights major drawbacks of deep-learning methods, i.e., it is difficult to interpret the model and to explain classification criteria. It is worth mentioning that if the noise also were added to the training data, the classifier might cater for this and be more robust against such changes.

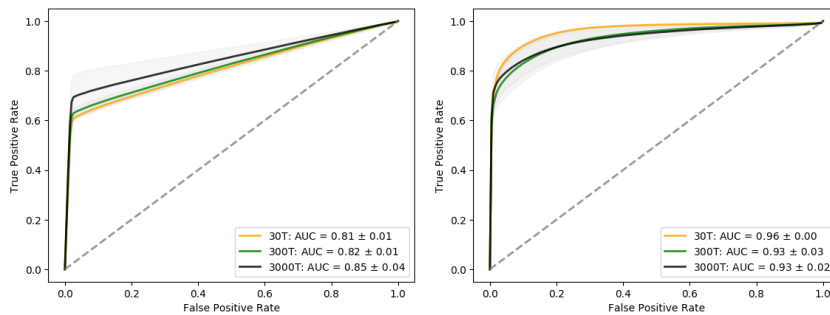


Figure 11. (Left panel) ROC curve with Gaussian noise is simulated with a standard deviation of 0.01 is added to the test data set. The drop in accuracy is quite large, even with relatively low level of noise added to the test data. (Right Panel) ROC curve for the case where we have excluded the 300T scenario.

3.7. Making Decisions Based on BCNN Output with Varying Cost

In Section 2.7 we presented Algorithm 1 that can assist in taking difficult decisions during environmental monitoring. The remainder of this Section exemplifies the use of Algorithm 1, given a specific cost function. We have created $T = 200$ realizations as an approximation of the two posterior distributions $P_i(c_1|\mathbf{x}^*)$ and $P_i(c_2|\mathbf{x}^*)$. Since we use binary classification, we have that $P_i(c_2|\mathbf{x}^*) = 1 - P_i(c_1|\mathbf{x}^*)$. The 200 realization can be viewed as distinct expert opinions of what class the time series belongs. By applying the two summary function Equations (5) and (6) to the posterior predictive distribution we get the following decision rule in the binary classification setting, that is, we decide action a_1 if

$$(\lambda_{21} - \lambda_{11})P(c_1|\mathbf{x}^*) > (\lambda_{12} - \lambda_{22})P(c_2|\mathbf{x}^*).$$

Since it is difficult to estimate the real cost associated with confirming a leakage, we use a simple cost function and vary some parameters to determine the impact of the cost function on the optimal decision. The cost associated with confirming a no-leak situation is the baseline cost, i.e., we assign a value of 1 for this purpose. This cost can be interpreted as the operational cost with sending a cruise or initiate a unmanned operation to check for a leakage. We assume that the cost with confirming a leakage is dependent on the operational cost. Furthermore, the cost of not confirming a leakage is κ times more expensive than the operational cost. The parameter γ is the factor that describes the cost of not confirming a leakage compared to the cost of confirming a leakage. The cost associated with not confirming if there is no leakage is assumed to be 0. Based on these assumptions we can set up a decision rule dependent on the summary function over the posterior distribution; confirm a leak if

$$(\gamma - 1)\kappa > \frac{P(c_1|\mathbf{x}^*)}{P(c_2|\mathbf{x}^*)}. \quad (7)$$

Table 3 shows the cost function in terms of true and false positives and true and false negatives.

Table 3. Overview of the cost functions applied in this example. The cost function can be interpreted as having no cost to doing the correct decision. Confirming a no-leak has some operational cost, e.g., a ship must mobilize and search/confirm that there is no leakage. Confirm a leakage has a cost of κ times the operational cost. The cost of not confirming a true leakage is gamma times the cost of confirming a leakage. We assume there is no cost with not looking for a leak when there is no leakage.

	Leak	No-Leak
Confirm (α_1)	$\lambda_{11} = \kappa$	$\lambda_{12} = 1$
Not Confirm (α_2)	$\lambda_{21} = \gamma\lambda_{11}$	$\lambda_{22} = 0$

In Figure 12 we show the percentage of the total area to be monitored for $\kappa \in \{10, 100, 1000\}$ and $1 \leq \gamma \leq 100$.

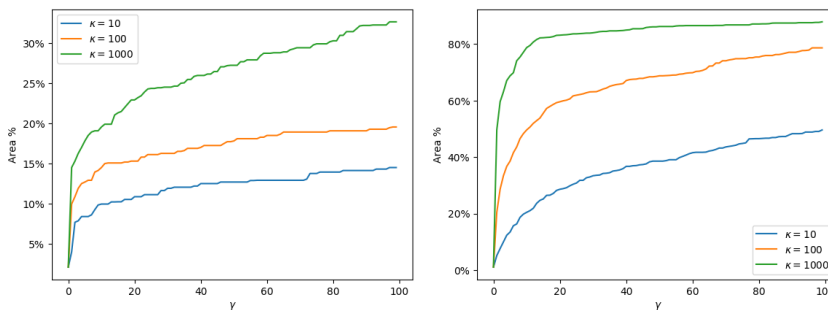


Figure 12. The figure shows the percentage of the total area to be monitored where the optimal decision would be confirming a leak. The cost function is altered by varying the parameter κ and γ in Equation (7) and Table 3. The three different lines represents varying κ , and the the x-axis shows varying γ for each scenario. Increased cost difference between the operational cost and the cost of confirming a leak, result in a higher degree of confirmation, faster. (Left Panel) The MAP of the predictive posterior distribution. (Right Panel) The expectation of the predictive posterior distribution. Using the mode instead of the expectation results in less confirmation/mobilization.

This cost function shows the trade-off between confirming a leak and not confirming a leak. By using the MAP, the predictive distribution becomes more distinct on which class it belongs. We see this in Figure 12, where the percentage area needed to be monitored under the same cost function is significantly lower.

4. Discussion

We have presented a three-step algorithm that combines BCNNs with Bayesian decision theory, for support in environmental monitoring. In the first step a BCNN is optimize on labeled simulated data. In the next step, the BCNN is used to generate a posterior predictive distribution of class labels. In the final step, the optimal monitoring strategy is calculated based on the posterior predictive distribution and given operational costs.

Our case study indicates that MC dropout and BCNN are effective tools for detecting CO₂ seeps in marine waters. The overall predictive power is large, as seen through the ROC curve in Figure 4. While the majority of the time series are either classified as leak or no-leak with little uncertainty, a small proportion will be inconclusive; see Figure 5. As expected, the classification uncertainties increase with the distance to the leak source location. This distance depends on the seep flux rates and can be modeled from the outcome of the BCNN; see Figure 6.

Demanding high detection probability reduces the detectable area of a monitoring location, i.e., you will need to measure closer to a leak in order to achieve the desired accuracy; see Figure 10. This motivates the use of optimal decision strategies, that is finding appropriate action that should be initiated based on the prediction, its uncertainty, and costs associated with taking wrong and right decisions. We gave an example of an optimal strategy choice in a binary classification setting in Section 3.7 for different costs, showcasing the algorithm.

There is extensive literature, including several deep-learning approaches, on monitoring and forecasting of air quality time series. For example Biancofiore et al. [35] used a recursive and feed forward neural network and Freeman et al. [36] a RNN to forecast and monitor air quality from time series data. In context of oceanography monitoring, Bezdek et al. [67] used hyper-ellipsoidal models for detecting anomalies in a wireless sensor network and validated their approach on data from Great Barrier Reef. None of these approaches take uncertainty into account. Ahmad [68] recently published a literature review of machine-learning applications in oceanography.

As CO₂ is naturally present in the ocean and is highly variable, it is difficult to attribute measured CO₂ to an unplanned release. Blackford et al. [39] developed pH anomaly criterion for determination of marine CO₂ leaks. This criterion is however location dependent and cannot be generalized. In contrast to Blackford et al., our method automatically extracts features and characteristics that could be hidden within the natural variability, potentially increasing the detection probability. Due to the costly survey in case of confirmation of leakages, it will be of importance with information about the model's uncertainty, our approach provides that.

The most important strength of our approach is that it is principled and consistently uses a probabilistic Bayesian framework for effective decision-making in the context of environmental monitoring. A potential weakness of our case study is that we did not systematically optimize hyper-parameters, which could potentially improve the results. Examples are increasing or decrease the number of CNN layers, changing activation functions, loss functions, optimization procedures, and size of filters, strides and kernel size of the convolutional layers. With respect to loss function and activation function, we have chosen the standard techniques used in the machine-learning community today and have not addressed these norms. With respect to size of filters, strides and kernel size of the convolutional layers, we have only performed a small trial and error search of the possible hyper-parameters. Looking into the model's weights shows that quite a large part of the network is active, suggesting that the model structure and size is relatively well-balanced. We have not found it feasible to use a lot of time to tune the model further, as we have achieved good results with the current hyper-parameter configuration.

Another more serious limitation is the fact that the BCNN is optimized with data based on a limited number of simulations for a limited time period. This biases the BCNN model towards the simulated conditions. The model can still be used in a general scenario, but its predictive power will decrease. That is why more simulations with different forcing, leak locations and fluxes are necessary to generate a predictive model that is more robust.

To test how well our BCNN model generalizes, we have carried out some sensitivity analyses in Section 3.6. The general observation was that adding noise to the test data increased uncertainty and decreased predictive accuracy. With low level of noise the prediction deteriorates, mainly because no-leak time series is miss-classified as leaks, false positives. However, it is still able to distinguish the two classes with relatively good success. Corruption of deep-learning models is a well know problem, even with small alterations in the input data. This problem might have been solved by training the model on noisy data.

A second sensitivity analysis showed that removal of the 300T data set reduced the overall predictive power in all cases; however, the method was still able with high confident and accuracy to predict the 300T test data set. This indicates at least good generalization properties to different leakage fluxes, and potentially to different leakage locations.

In this study, univariate time series was used; however, if more information were available in terms of different sensors, measuring different geochemical markers, this framework could be extended to a multi-variate TSC setting. Another extension could be to increase the number of classes and treat them as a multi-label classification tasks, with the purpose to classify time series as be either of the following classes; 0T, 30T, 300T or 3000T.

Another question we would like to study is how well our model generalizes to different locations. One of the key concepts of CNNs is the ability to extract important features from the data. Here the concept capability to capture key characteristics of time series that contain a leak signal or not. In this sense, the model should generalize well to other locations. Testing this hypothesis would be an important step towards wide-spread use of our method.

However, in our view, the most interesting extension would be to extend the framework to a transfer learning setting. The concept of transfer learning is to use a pre-trained model and fix the weights of the first few layers and re-train with an entirely new data set. Transfer learning has been used with success in computer vision and other tasks, and the key benefit is that the need for data is reduced drastically. Translation of this concept to binary TSC, where the pre-trained model would be trained on simulation data, should be investigated closer.

Author Contributions: Conceptualization, K.G.; methodology, K.G., A.O. and N.B.; software, K.G.; validation, K.G.; formal analysis, K.G., N.B. and A.O.; investigation, K.G.; data curation, K.G.; writing—original draft preparation, K.G., A.O. and N.B.; writing—review and editing, K.G., A.O., N.B. and G.A.; visualization, K.G.; supervision, G.A., N.B., A.O., Resources; G.A.; Funding acquisition, G.A.; Project administration, K.G. and G.A. All authors have read and agreed to the published version of the manuscript.

Funding: This project has received funding from the European Unions’s Horizon 2020 research and innovation programme under grant agreement No. 654462 and the Research Council of Norway (project no. 254711 BayMoDe). In addition, this work is part of the project ACTOM, funded through the ACT programme (Accelerating CCS Technologies, Horizon2020 Project No 294766). Financial contributions made from; The Research Council of Norway, (RCN), Norway, Netherlands Enterprise Agency (RVO), Netherlands, Department for Business, Energy & Industrial Strategy (BEIS) together with extra funding from NERC and EPSRC research councils, United Kingdom, US-Department of Energy (US-DOE), USA. In-kind contributions from the University of Bergen are gratefully acknowledged.

Acknowledgments: The authors would like to acknowledge NVIDIA Corporation for providing their GPUs in the academic GPU Grant Program. We would also acknowledge Pierre Cazenave, Plymouth Marine Laboratory, for providing the data set.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, in the analysis, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

AUC	Area Under the Curve
BCNN	Bayesian Convolutional Neural Network
CCS	Carbon Capture and Storage
CO ₂	Carbon dioxide
CNN	Convolutional Neural Network
DOAJ	Directory of open access journals
DTW	Dynamic Time Warping
ERSEM	European Regional Seas Ecosystem Model
FVCOM	Finite-Volume Community Model
IOC	Intergovernmental Oceanographic Commission
KDE	Kernel Density Estimate
MAP	Maximum A Posteriori
MDPI	Multidisciplinary Digital Publishing Institute
MC	Monte Carlo

MCMC	Markov Chain Monte Carlo
ReLU	REctified Linear Unit
RNN	Recurrent Neural Networks
ROC	Receiver Operating Characteristic
STEMM-CCS	Strategies for Environmental Monitoring of Marine Carbon Capture and Storage
TSC	Time Series Classification
UN	United Nations

References

- Halpern, B.S.; Longo, C.; Hardy, D.; McLeod, K.L.; Samhuri, J.F.; Katona, S.K.; Kleisner, K.; Lester, S.E.; O’Leary, J.; Ranelletti, M.; et al. An index to assess the health and benefits of the global ocean. *Nature* **2012**, *488*, 615–620. [CrossRef] [PubMed]
- Domínguez-Tejo, E.; Metternicht, G.; Johnston, E.; Hedge, L. Marine Spatial Planning advancing the Ecosystem-Based Approach to coastal zone management: A review. *Mar. Policy* **2016**, *72*, 115–130. [CrossRef]
- Ali, A.; Thiem, Ø.; Berntsen, J. Numerical modelling of organic waste dispersion from fjord located fish farms. *Ocean Dyn.* **2011**, *61*, 977–989. [CrossRef]
- Hylland, K.; Burgeot, T.; Martínez-Gómez, C.; Lang, T.; Robinson, C.D.; Svavarsson, J.; Thain, J.E.; Vethaak, A.D.; Gubbins, M.J. How can we quantify impacts of contaminants in marine ecosystems? The ICON project. *Mar. Environ. Res.* **2017**, *24*, 2–10. [CrossRef] [PubMed]
- First, P.J. Global Warming of 1.5 °C An IPCC Special Report on the Impacts of Global Warming of 1.5 °C Above Pre-Industrial Levels and Related Global Greenhouse Gas Emission Pathways, in the Context of Strengthening the Global Response to the Threat of Climate Change. *Sustain. Dev. Efforts Eradicate Poverty* **2019**, *1*, 1–22.
- Agency, I.E. *Global Energy & CO₂ Status Report*; Technical Report; IEA: Paris, France, 2018.
- Bauer, S.; Beyer, C.; Dethlefsen, F.; Dietrich, P.; Duttmann, R.; Ebert, M.; Feeser, V.; Görke, U.; Köber, R.; Kolditz, O.; et al. Impacts of the use of the geological subsurface for energy storage: An investigation concept. *Environ. Earth Sci.* **2013**, *70*, 3935–3943. [CrossRef]
- Blackford, J.; Bull, J.M.; Cevatoglu, M.; Connelly, D.; Hauton, C.; James, R.H.; Lichtschlag, A.; Stahl, H.; Widdicombe, S.; Wright, I.C. Marine baseline and monitoring strategies for carbon dioxide capture and storage (CCS). *Int. J. Greenh. Gas Control* **2015**, *38*, 221–229. [CrossRef]
- Jones, D.; Beaubien, S.; Blackford, J.; Foekema, E.; Lions, J.; Vittor, C.D.; West, J.; Widdicombe, S.; Hauton, C.; Queirós, A. Developments since 2005 in understanding potential environmental impacts of [CO₂] leakage from geological storage. *Int. J. Greenh. Gas Control* **2015**, *40*, 350–377. [CrossRef]
- Yang, Q.; Wu, X. 10 challenging problems in data mining research. *Int. J. Inf. Technol. Decis. Mak.* **2006**, *5*, 597–604. [CrossRef]
- Esling, P.; Agon, C. Time-series data mining. *ACM Comput. Surv. (CSUR)* **2012**, *45*, 12. [CrossRef]
- Bagnall, A.; Lines, J.; Bostrom, A.; Large, J.; Keogh, E. The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances. *Data Min. Knowl. Discov.* **2017**, *31*, 606–660. [CrossRef] [PubMed]
- Berndt, D.J.; Clifford, J. *Using Dynamic Time Warping to Find Patterns in Time Series*; KDD Workshop: Seattle, WA, USA, 1994; Volume 10, pp. 359–370.
- Ratanamahatana, C.A.; Keogh, E. Three myths about dynamic time warping data mining. In Proceedings of the 2005 SIAM International Conference on Data Mining, SIAM, Newport Beach, CA, USA, 21–23 April 2005; pp. 506–510.
- Bagnall, A.; Janacek, G. A run length transformation for discriminating between auto regressive time series. *J. Classif.* **2014**, *31*, 154–178. [CrossRef]
- Smyth, P. Clustering Sequences with Hidden Markov Models. Available online: <http://papers.nips.cc/paper/1217-clustering-sequences-with-hidden-markov-models.pdf> (accessed on 1 June 2020).
- Williams, C.K.; Barber, D. Bayesian classification with Gaussian processes. *IEEE Trans. Pattern Anal. Mach. Intell.* **1998**, *20*, 1342–1351. [CrossRef]
- James, G.M.; Hastie, T.J. Functional linear discriminant analysis for irregularly sampled curves. *J. R. Stat. Soc. Ser. B* **2001**, *63*, 533–550. [CrossRef]

19. Hall, P.; Poskitt, D.S.; Presnell, B. A functional data—Analytic approach to signal discrimination. *Technometrics* **2001**, *43*, 1–9. [[CrossRef](#)]
20. Bagnall, A.; Lines, J.; Hills, J.; Bostrom, A. Time-series classification with COTE: The collective of transformation-based ensembles. *IEEE Trans. Knowl. Data Eng.* **2015**, *27*, 2522–2535. [[CrossRef](#)]
21. Lines, J.; Taylor, S.; Bagnall, A. Time Series Classification with HIVE-COTE: The Hierarchical Vote Collective of Transformation-Based Ensembles. *ACM Trans. Knowl. Discov. Data* **2018**, *12*. [[CrossRef](#)]
22. Fawaz, H.I.; Forestier, G.; Weber, J.; Idoumghar, L.; Muller, P.A. Deep learning for time series classification: A review. *Data Min. Knowl. Discov.* **2019**, *33*, 1–47.
23. Zheng, Y.; Liu, Q.; Chen, E.; Ge, Y.; Zhao, J.L. Time series classification using multi-channels deep convolutional neural networks. In Proceedings of the International Conference on Web-Age Information Management, Macau, China, 16–18 June 2014; pp. 298–310.
24. Wang, Z.; Yan, W.; Oates, T. Time series classification from scratch with deep neural networks: A strong baseline. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 1578–1585.
25. Hüsken, M.; Stagge, P. Recurrent neural networks for time series classification. *Neurocomputing* **2003**, *50*, 223–235. [[CrossRef](#)]
26. MacKay, D.J.C. A Practical Bayesian Framework for Backpropagation Networks. *Neural Comput.* **1992**, *4*, 448–472. [[CrossRef](#)]
27. Alendal, G.; Blackford, J.; Chen, B.; Avlesen, H.; Omar, A. Using Bayes Theorem to Quantify and Reduce Uncertainties when Monitoring Varying Marine Environments for Indications of a Leak. *Energy Procedia* **2017**, *114*, 3607–3612. [[CrossRef](#)]
28. Gal, Y.; Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1050–1059.
29. Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; Wierstra, D. Weight uncertainty in neural networks. *arXiv* **2015**, arXiv:1505.05424.
30. Shridhar, K.; Laumann, F.; Liwicki, M. A comprehensive guide to bayesian convolutional neural network with variational inference. *arXiv* **2019**, arXiv:1901.02731.
31. Leibig, C.; Allken, V.; Ayhan, M.S.; Berens, P.; Wahl, S. Leveraging uncertainty information from deep neural networks for disease detection. *Sci. Rep.* **2017**, *7*, 17816. [[CrossRef](#)] [[PubMed](#)]
32. Abideen, Z.U.; Ghafoor, M.; Munir, K.; Saqib, M.; Ullah, A.; Zia, T.; Tariq, S.A.; Ahmed, G.; Zahra, A. Uncertainty Assisted Robust Tuberculosis Identification With Bayesian Convolutional Neural Networks. *IEEE Access* **2020**, *8*, 22812–22825. [[CrossRef](#)]
33. Kendall, A.; Cipolla, R. Modelling uncertainty in deep learning for camera relocalization. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 4762–4769.
34. Malde, K.; Handegard, N.O.; Eikvil, L.; Salberg, A.B. Machine intelligence and the data-driven future of marine science. *ICES J. Mar. Sci.* **2019**. [[CrossRef](#)]
35. Biancofiore, F.; Busilacchio, M.; Verdecchia, M.; Tomassetti, B.; Aruffo, E.; Bianco, S.; Di Tommaso, S.; Colangeli, C.; Rosatelli, G.; Di Carlo, P. Recursive neural network model for analysis and forecast of PM10 and PM2.5. *Atmos. Pollut. Res.* **2017**, *8*, 652–659. [[CrossRef](#)]
36. Freeman, B.S.; Taylor, G.; Gharabaghi, B.; Thé, J. Forecasting air quality time series using deep learning. *J. Air Waste Manag. Assoc.* **2018**, *68*, 866–886. [[CrossRef](#)] [[PubMed](#)]
37. Kiiveri, H.; Caccetta, P.; Campbell, N.; Evans, F.; Furby, S.; Wallace, J. Environmental Monitoring Using a Time Series of Satellite Images and Other Spatial Data Sets. In *Nonlinear Estimation and Classification*; Denison, D.D., Hansen, M.H., Holmes, C.C., Mallick, B., Yu, B., Eds.; Springer: New York, NY, USA, 2003; pp. 49–62. [[CrossRef](#)]
38. Banskota, A.; Kayastha, N.; Falkowski, M.J.; Wulder, M.A.; Froese, R.E.; White, J.C. Forest Monitoring Using Landsat Time Series Data: A Review. *Can. J. Remote Sens.* **2014**, *40*, 362–384. [[CrossRef](#)]
39. Blackford, J.; Artioli, Y.; Clark, J.; de Mora, L. Monitoring of offshore geological carbon storage integrity: Implications of natural variability in the marine system and the assessment of anomaly detection criteria. *Int. J. Greenh. Gas Control* **2017**, *64*, 99–112. [[CrossRef](#)]

40. Siddorn, J.R.; Allen, J.I.; Blackford, J.C.; Gilbert, F.J.; Holt, J.T.; Holt, M.W.; Osborne, J.P.; Proctor, R.; Mills, D.K. Modelling the hydrodynamics and ecosystem of the North-West European continental shelf for operational oceanography. *J. Mar. Syst.* **2007**, *65*, 417–429. [[CrossRef](#)]
41. Alendal, G.; Drange, H. Two-phase, near-field modeling of purposefully released CO₂ in the ocean. *J. Geophys. Res. Ocean.* **2001**, *106*, 1085–1096. [[CrossRef](#)]
42. Dewar, M.; Sellami, N.; Chen, B. Dynamics of rising CO₂ bubble plumes in the QICS field experiment. *Int. J. Greenh. Gas Control* **2014**. doi:10.1016/j.ijggc.2014.11.003. [[CrossRef](#)]
43. Ali, A.; Froyso, H.G.; Avlesen, H.; Alendal, G. Simulating spatial and temporal varying CO₂ signals from sources at the seafloor to help designing risk-based monitoring programs. *J. Geophys. Res. Ocean.* **2016**, *121*, 745–757. [[CrossRef](#)]
44. Blackford, J.; Alendal, G.; Avlesen, H.; Brereton, A.; Cazenave, P.W.; Chen, B.; Dewar, M.; Holt, J.; Phelps, J. Impact and detectability of hypothetical CCS offshore seep scenarios as an aid to storage assurance and risk assessment. *Int. J. Greenh. Gas Control* **2020**, *95*, 102949. [[CrossRef](#)]
45. Vielstädte, L.; Karstens, J.; Haeckel, M.; Schmidt, M.; Linke, P.; Reimann, S.; Liebetrau, V.; McGinnis, D.F.; Wallmann, K. Quantification of methane emissions at abandoned gas wells in the Central North Sea. *Mar. Pet. Geol.* **2015**, *68*, 848–860. [[CrossRef](#)]
46. Pan, S.J.; Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **2009**, *22*, 1345–1359. [[CrossRef](#)]
47. Chen, C. *An Unstructured-Grid, Finite-Volume Community Ocean Model: FVCOM User Manual*; Sea Grant College Program, Massachusetts Institute of Technology: Cambridge, MA, USA, 2012.
48. Butenschön, M.; Clark, J.; Aldridge, J.N.; Allen, J.I.; Artioli, Y.; Blackford, J.; Bruggeman, J.; Cazenave, P.; Ciavatta, S.; Kay, S.; et al. ERSEM 15.06: A generic model for marine biogeochemistry and the ecosystem dynamics of the lower trophic levels. *Geosci. Model Dev.* **2016**, *9*, 1293–1339.
49. Hähnel, P.; Mareček, J.; Monteil, J.; O'Donncha, F. Using deep learning to extend the range of air pollution monitoring and forecasting. *J. Comput. Phys.* **2020**, *408*, 109278. [[CrossRef](#)]
50. Ruthotto, L.; Haber, E. Deep neural networks motivated by partial differential equations. *J. Math. Imaging Vis.* **2019**, *62*, 1–13. [[CrossRef](#)]
51. Gal, Y. Uncertainty in Deep Learning. Ph.D. Thesis, University of Cambridge, Cambridge, UK, 2016.
52. Gal, Y.; Ghahramani, Z. Bayesian convolutional neural networks with Bernoulli approximate variational inference. *arXiv* **2015**, arXiv:1506.02158.
53. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
54. Tibshirani, R.J.; Efron, B. An introduction to the bootstrap. *Monogr. Stat. Appl. Probab.* **1993**, *57*, 1–436.
55. Berger, J.O. *Statistical Decision Theory and Bayesian Analysis*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013.
56. Cazenave, P.; Blackford, J.; Artioli, Y. Regional Modelling to Inform the Design of Sub-Sea Co₂ Storage Monitoring Networks. In Proceedings of the 14th Greenhouse Gas Control Technologies Conference Melbourne, Melbourne, Australia, 21–26 October 2018; pp. 21–26.
57. Riebesell, U.; Fabry, V.J.; Hansson, L.; Gattuso, J.P. *Guide to Best Practices for Ocean Acidification Research and Data Reporting*; Office for Official Publications of the European Communities: Brussels, Belgium, 2011.
58. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, 21–24 June 2010; pp. 807–814.
59. Baldi, P.; Sadowski, P. The dropout learning algorithm. *Artif. Intell.* **2014**, *210*, 78–122. [[CrossRef](#)]
60. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
61. Hvidevold, H.K.; Alendal, G.; Johannessen, T.; Ali, A.; Mannseth, T.; Avlesen, H. Layout of CCS monitoring infrastructure with highest probability of detecting a footprint of a CO₂ leak in a varying marine environment. *Int. J. Greenh. Gas Control* **2015**, *37*, 274–279. [[CrossRef](#)]
62. Greenwood, J.; Craig, P.; Hardman-Mountford, N. Coastal monitoring strategy for geochemical detection of fugitive CO₂ seeps from the seabed. *Int. J. Greenh. Gas Control* **2015**, *39*, 74–78. [[CrossRef](#)]
63. Hvidevold, H.K.; Alendal, G.; Johannessen, T.; Ali, A. Survey strategies to quantify and optimize detecting probability of a CO₂ seep in a varying marine environment. *Environ. Model. Softw.* **2016**, *83*, 303–309. [[CrossRef](#)]

64. Alendal, G. Cost efficient environmental survey paths for detecting continuous tracer discharges. *J. Geophys. Res. Ocean.* **2017**, *122*, 5458–5467. [[CrossRef](#)]
65. Oleynik, A.; García-Ibáñez, M.I.; Blaser, N.; Omar, A.; Alendal, G. Optimal sensors placement for detecting CO₂ discharges from unknown locations on the seafloor. *Int. J. Greenh. Gas Control* **2020**, *95*, 102951. [[CrossRef](#)]
66. Botnen, H.; Omar, A.; Thorseth, I.; Johannessen, T.; Alendal, G. The effect of submarine CO₂ vents on seawater: Implications for detection of subsea Carbon sequestration leakage. *Limnol. Oceanogr.* **2015**, *60*, 402–410. [[CrossRef](#)]
67. Bezdek, J.C.; Rajasegarar, S.; Moshtaghi, M.; Leckie, C.; Palaniswami, M.; Havens, T.C. Anomaly detection in environmental monitoring networks [application notes]. *IEEE Comput. Intell. Mag.* **2011**, *6*, 52–58. [[CrossRef](#)]
68. Ahmad, H. Machine learning applications in oceanography. *Aquat. Res.* **2019**, *2*, 161–169. [[CrossRef](#)]



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).



Graphic design: Communication Division, UIB / Print: Skjipes Kommunikasjon AS



uib.no

ISBN: 9788230853269 (print)
9788230868843 (PDF)