Aalto University
School of Science
Master's Programme in Life Science Technologies

Bent Ivan Oliver Harnist

Probabilistic Precipitation Nowcasting using Bayesian Convolutional Neural Networks

Master's Thesis
Espoo, July 20th, 2022

DRAFT! — July 5, 2022 — DRAFT!

Supervisor: Professor Arno Solin
Advisor: Terhi Mäkinen D.Sc.
Seppo Pulkkinen D.Sc.

Aalto University
School of Science

Aalto University
School of Science
Master's Programme in Life Science Technologies

ABSTRACT OF
MASTER'S THESIS

Author:	Bent Ivan Oliver Harnist		
Title:	Probabilistic Precipitation Nowcasting using Bayesian Convolutional Neural Networks		
Date:	July 20th, 2022	**Pages:**	52
Major:	Complex Systems	**Code:**	SCI3060
Supervisor:	Professor Arno Solin		
Advisor:	Terhi Mäkinen D.Sc. Seppo Pulkkinen D.Sc.		
!FIXME **This is an example how to use fixme: add your abstract here.** FIXME!			
Keywords:	Precipitation nowcasting,		
Language:	English		

Aalto-yliopisto
Perustieteiden korkeakoulu
Tieto-, tietoliikenne- ja informaatiotekniikan maisteriohjelma

DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Bent Ivan Oliver Harnist		
Työn nimi:	Probabilistinen Sateen Nowcasting käyttäen Bayesilaisia Konvolutiivisia Neuroverkkoja		
Päiväys:	20. heinäkuuta 2022	**Sivumäärä:**	52
Pääaine:	Complex Systems	**Koodi:**	SCI3060
Valvoja:	Professori Arno Solin		
Ohjaaja:	Terhi Mäkinen D.Sc. Seppo Pulkkinen D.Sc.		
joku			
Asiasanat:	Sateen ennustaminen,		
Kieli:	Englanti		

Acknowledgements

!Fixme **My acknowledgements** Fixme!

Espoo, July 20th, 2022

Bent Ivan Oliver Harnist

Abbreviations and Acronyms

DL	Deep learning
NWP	Numerical Weather Prediction
BNN	Bayesian Neural Network
BRN	Bayesian RainNet
STEPS	Short-Term Ensemble Prediction System
LINDA	Lagrangian Integro-Difference equation model with Autoregression
Z	Reflectivity factor
dBZ	Decibel relative to Z
ELBO	Evidence Lower Bound
CSI	Critical Success Index
ETS	Equitable Threat Score
FAR	False Alarm Rate
POD	Probability of Detection
MAE	Mean Absolute Error
ME	Mean Error
FSS	Fractions Skill Score
RAPSD	Radially-averaged Power Spectral Density
CRPS	Continuous Ranked Probability Score
ROC	Receiver Operating Characteristic
WMO	World Meteorological Organization

Contents

Chapter 1

Introduction

Nowcasting is defined as weather forecasting at a local scale up to six hours, according the World Meteorological Organization (WMO) definition [44]. The ability to provide an accurate precipitation nowcast has grown during this century into a meteorologically and societally significant challenge. This significance emanates from the fact that early warnings of severe precipitation enable authorities and other actors to make early decisions enabling for example disaster damage control, traffic safety, and in the case of heavy rainfall flash flood prevention, as well as the mitigation of economic loss incurred by them. High-density urbanization has exacerbated these issues, making it evermore vital to discover reliable and skillful methods for precipitation nowcasting.

Numerical weather prediction (NWP) solves the partial differential equations governing the physical processes of weather and climate to produce forecasts. NWP has seen great improvements over decades, up to the point where it can be used to produce accurate forecasts for up to a week in some cases [8]. However, such method is not applicable to predicting at timescales as short as 0 to 6 hours. This is mainly due to imperfect initialization making simulations unable to reach numerical stability in such short time timescales. Other problems with NWP include its computational expensiveness and generally having insufficient spatiotemporal to predict convective events and heavy localized rainfall [46].

To compensate for the shortcomings of NWP in the realm of precipitation nowcasting, many different dedicated models and algorithms have been developed [36]. Contrary to NWP, these models do not combine data from multiple sources in making predictions, but are simpler in a way as they use only a single input field. In practice, many of those systems are based on the estimation of the precipitation advection field from radar echo image sequences and the further extrapolation of these sequences along the advec-

tion field [36, 40]. Other methods have been developed that track and try to nowcast the evolution of individual rain cells instead of the whole grid [17, 36].

Dedicated nowcasting methods have had great success in forecasting the immediate future, but their performance typically degrades quickly, becoming unreliable often in the range of an hour. This is related to the fact that basic extrapolation-based methods have limitations such as failing to capture nonlinear patterns like growth and decay of precipitation, as well as convective cell initiation and their life-cycle.

A lot of work has been done in nowcasting in trying to incorporate modeling of higher-order and multiscale phenomena into advection-extrapolation based models. Recently, Deep-learning (DL) based precipitation nowcasting approaches have started showing promising results delving into the issues of traditional methods thanks to the high volume of training data available, the increase in computational power, the expressivity of those models and their relative cheapness of inference [7, 48, 49].

1.1 Problem statement

While the above introduction has cast the problem of nowcasting as finding a single optimal point estimate for future precipitation, it is useful to think about it in a probabilistic way. Because in nowcasting the most interesting phenomena are extreme events, which are those requiring preparation and early warnings, it makes intuitive sense that accurate and reliable probabilistic nowcasts are primordial for operational decision-making in meteorological crises. Also, because smaller spatial scales are less predictable, Adding a degree of uncertainty improves the model usefulness at those scales [22].

Indeed, the interest in probabilistic nowcasts has grown lately and many probabilistic models have been introduced in the last decades [5, 20, 45, 47], notably the Short-Term Ensemble Prediction System (STEPS) [12] and more recently the Lagrangian Integro-Difference equation model with Autoregression (LINDA) [37]. These methods predict ensembles, that are used to estimate underlying distributions of data and precipitation probabilities. They perform reasonably well, but suffer from the same limitations as other extrapolation based methods, namely the inability to predict nonlinear patterns of growth and decay.

So far, only little work has been done on using DL to produce probabilistic precipitation nowcasts, with the biggest breaktrough perhaps being Ravuri et al. [39] using adversarially trained deep generative models to produce ensemble nowcasts. There exists many possible ways of making probabilistic

nowcasts using deep learning, with most of them focusing on directly modeling probability distributions of data. Another approach, that will be focused on from now on is instead to model the uncertainty of model parameters rather than that of the data, as would be done with STEPS or LINDA. This is done by using Bayesian Neural Networks and has an advantage of providing implicit regularization of model parameters thus reducing overfitting, while outputting an ensemble of predictions whose variability in part reflects network parameter uncertainty given the training data. Bayesian neural networks have been proposed for use in other risk-averse application such as biomedical image segmention [29] and autonomous vehicles [32].

In this work, the problem of making skillful and reliable probabilistic precipitation nowcasts will approached by building an uncertainty-aware Neural Network. A baseline Convolutional Neural Network (CNN) will be turned into a Bayesian Convolutional Network (BCNN) with optimization performed with stochastic variational inference (SVI) over model parameters. The model is selected, trained , and verified using Finnish Meteorological Institute (FMI) radar reflectivity composites. For the verification, both deterministic and probabilistic metrics are calculated in order to assess prediction skill against other probabilistic models and deterministic counterparts.

1.2 Structure of the Thesis

The present work is organized as follows. Chapter 2 contains background and a literature review on precipitation nowcasting and bayesian deep learning, aiming to familiarize the reader with essential concepts regarding the subject. Chapter 3 describes the experimental details of the work performed, including the datasets used for training and verification, the models implemented, as well as verification methods and baselines.

Chapter 4 presents the results of the experiments performed, including nowcasting examples of meteorologically interesting events, prediction skill measured with both deterministic and probabilistic metrics against multiple baseline models, as well as an assessment of predictive uncertainty. Chapter 5 discusses these results, their impact, and validity in details. Finally, Chapter 6 closes the thesis by summarizing the most important findings and takeaways.

Chapter 2

Background

2.1 Precipitation Nowcasting

Numerical Weather Prediction

Numerical weather prediction (NWP) is nowadays the main driver of operational weather prediction worldwide [8, 46]. On a very coarse level, NWP works by aggregating lots of meteorological observational data from sensors, which is used as initial state in solving atmospheric equations. This data can come from *in situ*, a.k.a. close contact ground-based sensors such as rain gauges and thermometers, or upper atmosphere measurements such as those from radio sondes or aircraft sensors. Additionally and most importantly, remote-sensing sources such as satellites and weather radars provide lots of observational data, and their efficient use is one of the reasons for the improvements of NWP. Then, data assimilation is performed with that data, filling in gaps and agglomerating different sources to be consistent with each others, making the output of this process appropriate to be fed as input into a numerical simulation of the atmosphere. These simulations then consist of solving Navier-Stokes equations multiple timesteps into the future. With the model output in hand, different post-processing can be applied to answer various questions about the future state of weather, such as making precipitation forecasts.

NWP can be performed on multiple spatio-temporal scales. Spatiotemporally coarse models cover a wide, sometimes global area and usually make simulations with higher *leadtimes*, meaning at further timesteps in the future. On the other hand, other models are specialized in providing finer spatiotemporal details with a smaller spatial extent. One example of this type of model is the High-Resolution Rapid Refresh (HRRR) model [4], developed by the United States National Oceanic and Atmospheric Administration (NOAA).

HRRR covers the continental united states, and has 3 kilometer spatial and 1 hour temporal resolution. Although very useful for many purposes, even such fine-detailed NWP models have not quite enough details to be most useful in very short term and very localized precipitation nowcasting as described in Chapter 1. As such, radars and radar-based nowcasting techniques shall be now described.

Weather Radars and Products

Precipitation nowcasting is largely based on the data collected from weather radars. There are many reasons for this, the main one being that radars are capable of directly observing precipitation particles (called hydrometeors) over a significant range with a high update rate and resolution [44]. This is something that no single other ground, upper-atmosphere, or radar observation is capable of providing, especially on the mesocscale ranging from tens to hundreds of kilometers. Combining many radars into a network can further increase sensing capacity to areas covering countries such as NEXRAD covering the continental united states [2], or entire continents such as the OPERA network covering most of Europe [43].

Radars were invented as a method to perform effective military surveillance during the second world war. Their potential use as weather observation tools was discovered coincidentally when it was realized that some echos of unknown origin in these surveillance radars were indeed precipitation. After the war, weather radars started to be routinely used in affluent parts of the world to provide precipitation observations and warnings.[18]

The working of weather radars is based on emitting a very strong, directed and polarized short electromagnetic pulse, which will interact with objects and particles in the atmosphere. This interaction is scattering, which reflects back a tiny fraction of the electromagnetic energy emitted at first. A receiver then listens to these returning signals, that are often called *radar echos*. A radar typically scans the atmosphere radially at multiple azimuthal elevation angles. In addition to that angle, the vertical path of the beams, and consequently the vertical position of objects detected, are affected by the curvature of the earth and the refractive index of the atmosphere decreasing with height, bending the beam downwards. [18]

The lowest elevation angle scan is the one giving the most accurate information about precipitation actually hitting the ground, with echos further away being less reliable, as they come from targets upper in the sky. Higher elevation angle scans or products derived from a multitude of those can be used to further inform about the dynamics and development of precipitation. Combining scans and derived products from multiple radars into a

single image makes what is called a *radar composite*.

In addition to meteorological echos, various non-meteorological sources are picked up by weather radars, such as solid obstacles, birds migrating, and insects. It is important to filter out or at least acknowledge those sources when preparing data for nowcasting.

Reflectivity (Z) is the quantity describing the amount of signal reflected from hydrometeors in the sky to the radar receiver. It is usually described in units of of decibel relative to Z, denoted dBZ. Standard weather radar beams are horizontally polarized, which produces the signals of interest when looking at radar scans for precipitation. One recent development of weather radars is the introduction of double polarization, meaning that in addition to the horizontal one, there are scans performed with vertically polarized beams. This enables estimation of precipitation type, drop shape, and helps with separating non-meteorological echos. Another important improvement to weather radars is the addition of doppler capacity, enabling estimation of target velocities from radar scans.

Converting the knowledge of (horizontal) reflectivity to that of rain rate is a highly non-trivial task and relies on approximations, because while water content is proportional to the third power of water droplet diameter, reflectivity of a single drop is proportional to its sixth power. Hence, total reflectivity depends on the drop size distribution, which depends on climatology and precipitation types. Given appropriate knowledge, the relationship between reflectivity and rain rate, commonly known as the Z-R relationship, is defined as

$$Z = AR^b \tag{2.1}$$

where Z is reflectivity, R is rainrate, and A as well as b are empirically determined parameters, usually by fitting rain gauge measurements to measured reflectivities. The most famous and widely used A and b parameters come from a study on drop size distribution from Marshal and Palmer in 1948 [31], where $A = 200$ and $b = 1.6$.

2.1.1 Classical deterministic methods

The reason for this is that extrapolation is based on the assumption of Lagrangian persistence, meaning that in Lagrangian coordinates, the field stays constant. In other

- Explain lagrangian frame of view vs eulerian
- Basic principles of advection equations.

- Chronological advancing, with +/- of each method
- Extrapolation: when introduced,
- TREC, COTREC, etc
- Burgers' eq, NS improvements
- S-PROG and predictability scale dependence by Germann
- ANVIL ...
- Cell based : TITAN and further improvements

2.1.2 Classical probabilistic methods

- Ensemble based, quantile based (?)
- STEPS, linked to S-PROG
- LINDA

2.2 Deep Learning for Nowcasting

2.2.1 Artificial Neural Networks

Artificial Neural Networks, abbreviated Neural Network or NN in a machine learning context, are computational systems inspired by biological neural networks, such as those present in the human brain. In this work, the terms Deep Learning will be used as meaning "regarding neural networks and their usage". Neural networks serve to accomplish many tasks, usually of the domain of artificial intelligence, such as regression, classification, or agent decision making. NN consist of discrete units called neurons linked to each others, usually arranged in functional units known as layers, in which case connections are formed between layers. Input data is fed into the NN to an input layer, transforming the input through learnable parameters, and this transformed signal is propagated through other layers further transforming the signal. The organization of layers and the way they are connected is known as the *network architecture*. Finally, signals converge to an output layer giving the product of the network, such as a prediction or class of input data. Between neurons and layers there are non-linear *activation functions*, which are in essence non-linear transformation on data, which are the basic mechanism that enables neural networks to learn complex nonlinear patterns.

Neural Network Optimization

In Neural Networks, parameters are usually not tuned by hand due to the sheer complexity of the task. Instead, finding optimal parameters for performing the task is framed as an optimization problem. This is accomplished using standard unconstrained optimization tools and the Backpropagation algorithm, allowing learning of the parameters, despite of the problem being very high-dimensional and non-convex. This algorithm is based on producing an output by passing input data through the network, and then calculating a metric of how well the network succeeded at the task, known as a *loss function* or cost function. The gradient of the output related to network parameters are then calculated backwards starting from the loss function, flowing through the network using the chain rule of derivation. These gradients are then used to update model parameters using simple gradient descent or its variations.

Usually, the input data is divided input small subsets called *batches* or mini-batches, and the gradient updates are calculated by going iteratively through them in a process called *training*. Gradient descent over these mini-batches is called Stochastic Gradient Descent. Going through all of the training data one time is called an *epoch*, and training is usually continued for many epochs. In deep learning, available data is usually split into training, validation, and test data. Validation data is used for the validation process, in which performance of the NN is assessed on data unused for training. This information on performance can then be used to tune *hyperparameters*, that is parameters that are constant and set before training, or perform other types decision-making related to the training process, such as early stopping of the training, if performance is likely not to improve anymore. The reason why training data is not used here is that the network usually performs better on data that it was trained on compared to other data. Lastly, test data is used for independent assessment of the network performance. This is needed because even though validation data is not used for training, the network is still biased to perform better on it if decision that improve performance were made based on it. Validation data can thus not be trusted on as an independent evaluator of network performance.

Different Varieties of Neural Networks

There are multiple types of Neural Network architectures that are each good at different types of task. Main types with some relevance to nowcasting and Bayesian Deep Learning are feed-forward neural networks, Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN) and Generative

Adversarial Networks (GAN). Feed-forward neural networks are the earliest type of NN, with linear transformations of data in layers of neurons, where neurons of successive layers are all connected to each others.

RNN are a modification of feed-forward neural networks with recurrent connections along units forming a network along a temporal sequence, making them widely used for the forecasting of time-series data and natural-language processing. Important improvements over original RNN are Long-Short Term Memory (LSTM) networks, with units having internal states allowing learning longer-range temporal dependencies, and Gated Recurrent Unit (GRU) modifying LSTM units to make them more lightweight.

CNN have layers that are convolutional filters applied over the input data. CNN are great for use in computer vision and other image related tasks, as they take advantage of the fact that learning short-range dependencies is enough to perform well in many of those tasks, thus reducing the hypothesis space of learnable representations in an informed way. One CNN architecture of particular interest in the context of this work is U-Net, that has shown excellent results in e.g. semantic segmentation tasks, that are important for biomedical imaging and autonomous vehicle applications. U-Net follows a simple encoder-decoder architecture. The next section will describe current applications of artificial neural networks to precipitation nowcasting.

2.2.2 Deep Learning approaches to Nowcasting

There exists now many methods for producing deterministic nowcasts with machine learning. Most suffer the same problem of producing overly blurry nowcasts only after a few timesteps. Despite there having been marginal improvement with regards to this problem over the years, but it remains one important challenge when it comes to producing deterministic nowcasts. The following chapters will present some of the main advancements in Deep Learning for nowcasting since its introduction.

Basic Approaches

The first dedicated Deep Learning model for precipitation nowcasting is ConvLSTM by Shi et al. [48], published in 2015, building upon the work of Oh et al. [34]. ConvLSTM is a neural network combining the image feature encoding capacities of CNN and temporal prediction capacities of LSTM into a single network fusing the two. ConvLSTM was shown to outperform an optical flow based model at predicting precipitation exceeding a low rain rate of 0.5 mm/h. A further model inspired by ConvLSTM called TrajGRU is developed by Shi et al. [49] in 2017 by replacing the LSTM component with

GRU and making recurrent connections dynamically location-variant. These connections improve the predictive skill over over invariant ones (ConvGRU), but the model was not tested against ConvLSTM.

Another class of models that have gained traction lately for nowcasting are simple U-Net based CNN, that adopts an image-to-image translation perspective on the problem. These networks are fed a sequence of radar images and output either one or several future frames. One example of this approach is that of Agrawal et al. [3]. In case only one frame is outputted, predictions for several leadtimes in the future may be done by iteratively feeding back predictions into the network as input, as is done with Ayzel et al.'s RainNet [7]. This iterative approach is more flexible but has the disadvantage of exacerbating the blurring problem. The CNN models in general have the advantage of being somewhat computationally less expensive when compared to ConvLSTM variants.

Notable Recent Improvements

Recently, Pan et al. improved the nowcasting of convective evolution by adding polarimetric variables (ZDR and KDP) in addition to reflectivity scans as inputs to a U-Net based model [35]. These polarimetric variables are derived from double polarization weather radars and their values are strongly associated with life-cycles of convective cells. This work showcases the importance of multichannel data and not only relying on reflectivity if one wants to accurately model nonlinear evolution of precipitation.

Above models suffers badly the the problem of blurring stated above, as do all current models based on discriminative neural networks, which are networks mapping one input to one output. Prediction blurring is in part related to loss functions used, as losses like ℓ_2 (Mean Squared Error (MSE)) and ℓ_1 tend to act that way when minimized with some uncertainty present. This excessive blurring also hinders the prediction of useful patterns such as heavy localized rainfall. Multi-Scale Structural Similarity Index (MS-SSIM), originally an image quality assessment metric [53], has recently started to be used as a loss function in deep learning, particularly in image reconstruction tasks. One of its main assets is that its single-scale counterpart (SSIM) has been shown to reduce blurring in image reconstruction [56] making it a good candidate loss function for nowcasting with neural networks. Indeed in recent years, there has been some cases where MS-SSIM or SSIM started have started to be used in Deep learning based nowcasting models, such as the one of Yin et al. (2021) [55]. Here both SSIM and MS-SSIM produced results surpassing MSE, with MS-SSIM giving the best results of the two. Most importantly, these loss functions seemed to better preserve high rain

rates, which tended to vanish with traditional losses at higher leadtimes.

Perhaps one of the biggest breakthroughs made yet, is the use of Generative Adversarial Networks (GAN) by Ravuri et al. last year [39]. This approach of using a generative model, i.e. one that generates samples from a probability distribution conditioned on past radar measurements, manages to solve the problem of nowcast blurring. GAN are a class of networks based on the competition between a generator network that generates the samples and discriminator(s) networks trying to discern whether these generated samples are real or fake. The generator then tries to fool the discriminators in a zero-sum game. Ravuri et al. use a generator network based on a convGRU architecture, nowcasting 90 minutes at once, two discriminators, and a regularization term penalizing deviations of generated samples at the grid level. The first discriminator ensures spatial consistency and the lack of blurring, while the second one ensures temporal consistency in generated sequences. The resulting generative model has slightly superior skill compared to existing approaches, but most importantly overwhelmingly over alternatives preserves features that make for a good nowcast from an operational forecaster point of view and is capable of providing probabilistic nowcasts which is very useful [39]

There has so far been limited work in probabilistic nowcasting with Deep Learning. In addition to the GAN by Ravuri et al. that can be used for such purpose, a probabilistic nowcasting model called MetNet was developed by Sonderby et al. [51]. The model is based on Axial Self-Attention by Ho et al. [25] and is capable of beating HRRR on probabilistic verification metrics for leadtimes of up to 8 hours.

2.3 Bayesian Deep Learning

This section gives an overview on Bayesian Neural Networks, that will be used make probabilistic nowcasts in this work.

2.3.1 Learning Probability Distributions

Neural networks are extremely useful model that on the other hand are very complex, in the sense that they have many optimizable parameters. The effect of this is that they are sensible to *overfitting*, meaning that if left unchecked they will learn spurious patterns in the data, over-adapting to the training dataset and worsening generalization ability. In order to counter this, different mechanisms exist that bias the neural network towards learning simpler representations that have better generalization ability when pre-

sented with out-of-training-data examples. This concept is known as *regularization*. Over the past decades, many regularization mechanisms have been successfully developed and applied to the training of deep neural networks. Notable examples include Dropout and weight decay, also known as L2-regularization.

One caveat of these classical regularization methods is that they do not enable representing the uncertainty of neural networks in unexplored regions, although they do improve performance in them. As it has understandably many benefits to make a neural network able to say "I don't know", a way to represent different plausible scenarios arising with novel data is needed.

There are two ways of approaching uncertainty estimation in neural networks. One is to directly model the uncertainty of predictions, and the other is to model the uncertainty of model parameters. These two are often complementary, but The latter approach has the benefit of providing implicit regularization to the network and is indeed the primary focus of this work. Learning this uncertainty of model parameters is most often accomplished using Bayesian Neural Networks (BNN). Strictly speaking, BNN are a class of stochastic neural networks, that is NN with stochastic components, where the parameters are probability distributions that are estimated using Bayesian inference.Bayesian Neural Networks in their current form were introduced by MacKay in 1992 [30], after early works starting in 1987 by Denker et al. [16], Tishby et al. [52], Denker and LeCun [15], and Buntine and Weigend [13].

In a Bayesian framework, the posterior distributions of parameters are estimated conditioned on data and a prior. Because exact Bayesian inference involves dealing with intractable integrals, it is not doable to solve them for real-world neural networks having thousands to millions of parameters. As such, two broad categories of inference methods have been developed for use in BNN. The first one is using Markov-Chain Monte-Carlo (MCMC) to sample from posterior distributions. MCMC works well for smaller-scale models, but it is limited to only thousands of parameters because of the computational expensiveness of Markov-Chain simulations.

The second inference method category is *Variational Inference* (VI). The main idea behind variational inference is to turn the problem of finding the intractable true bayesian posterior into that of finding the closest distribution from a limited distribution family (the variational family) with regards to the true posterior. This turns the problem of solving an integral into that of optimization, allowing the use of classical unconstrained optimization methods.

Dropout was surprisingly shown to be an instance of variational inference with a Bernoulli distributed posterior by Gal and Ghahramani in 2016 [21]. This legitimizes the use of Monte-Carlo Dropout, a technique used to get predictive uncertainty estimates from networks with Dropout layers, by simply

not switching off the layers for inference and thus drawing Monte-Carlo samples of the data distribution. Although not as expressive as explicit Bayesian NN, MC dropout has no computational overhead and is simple to implement, making it a very popular alternative to BNN.

In the context of Bayesian Deep Learning, the underlying neural network architecture will be referred to as the functional architecture or the functional model. Other components such as the inference method, the prior, or the posterior modeling will be commonly referred to as the stochastic model.

2.3.2 Variational inference

Variational inference as a means of training bayesian neural networks was first introduced by Hinton and Camp in 1993 [24]. However it was not used for a long time before breakthroughs by Graves et al. (2011) [23] and Blundell et al. (2015) [11] permitted its use in large-scale neural networks. The following sections will introduce the loss function and optimization algorithm used, mostly based on the work of Blundell et al. [11].

Evidence Lower Bound loss function

Finding the variational posterior $q(\boldsymbol{w}|\theta)$ best approximating the true posterior is formalized as minimizing the Kullback-Leiber (KL)-divergence between the two distributions as

$$\theta^* = \arg\min_{\theta} \mathrm{KL}(q(\boldsymbol{w}|\theta)||P(\boldsymbol{w}|\mathcal{D})) \tag{2.2}$$

where θ refers to variational posterior parameters,$\boldsymbol{w}$ to the weights, $\mathcal{D}$ to the data, q to the variational distribution, and P to the true distribution. The KL-divergence is defined as

$$KL((q(\boldsymbol{w}|\theta)||P(\boldsymbol{w}|\mathcal{D}))) = \int q(\boldsymbol{w}|\theta) \log \frac{q(\boldsymbol{w}|\theta)}{P(\boldsymbol{w}|\mathcal{D})} d\theta \tag{2.3}$$

This definition can be used to turn the minimization objective into

$$\arg\min_{\theta} \mathbb{E}_{q(\boldsymbol{w}|\theta)}[\log q(\boldsymbol{w}|\theta)] - \mathbb{E}_{q(\boldsymbol{w}|\theta)}[P(\boldsymbol{w}, \mathcal{D})] + \log P(\mathcal{D}) \tag{2.4}$$

Here the evidence $P(\mathcal{D})$ is very difficult to estimate. Luckily though, the minimization objective does not depend on it, so a new objective called the evidence lower bound (ELBO) or variational free energy is used instead, which is defined as

$$\mathrm{ELBO}(q) = -(KL((q(\boldsymbol{w}|\theta)||P(\boldsymbol{w}|\mathcal{D}))) - \log P(\mathcal{D})) \tag{2.5}$$

Solving for the evidence $\log P(\mathcal{D})$, it is clear that ELBO is a veritable lower bound for the it, as KL-divergences can not be negative.

ELBO can be rewritten as

$$\mathcal{F}(\mathcal{D}, \theta) = \mathrm{KL}[q(\boldsymbol{w}|\theta)||P(\boldsymbol{w})] - \mathbb{E}_{q(\boldsymbol{w}|\theta)}[\log(P(\mathcal{D}|\boldsymbol{w}))] \tag{2.6}$$

which is a form more suitable for the development of an optimization method. Here and from now on, ELBO will be referred to with the $\mathcal{F}$ symbol. For the rest of this work, the variational posterior family will be assumed to be that of Gaussian distributions, as the *Bayes-by-Backprop* (BBB) algorithm described in the following section works on this posterior family.

The Bayes-by-Backprop Algorithm

In order to apply backpropagation to Bayesian Neural Networks with variational inference, any sampling operation must be made external to the network in order to allow gradients to flow backwards. This is accomplished by parametrizing the standard deviation σ, which is the parameter linked to uncertainty in weights, with a parameter $\rho \in \mathbb{R}$, such that $\sigma = \log(1 + \exp(\rho)) \in (0, \infty)$. This is known as the reparametrization trick. Weights $\boldsymbol{w} = t(\sigma, \epsilon)$ are made deterministic functions of posterior parameters and the external stochastic noise ϵ. This is what allows gradients to flow through the network by making sampling external.

Using the reparametrization trick and the proposition

$$\frac{\partial}{\partial\theta}\mathbb{E}_{q(\boldsymbol{w}|\theta)}[f(\boldsymbol{w}, \theta)] = \mathbb{E}_{q(\epsilon)}[\frac{\partial f(\boldsymbol{w}, \theta)}{\partial \boldsymbol{w}} + \frac{\partial \boldsymbol{w}}{\partial \theta}\frac{\partial f(\boldsymbol{w}, \theta)}{\partial \boldsymbol{w}}] \tag{2.7}$$

from Blundell et al. [11], the ELBO cost function (Eq. 2.6) can be rewritten as in a form amendable to minibatch optimization, yielding

$$\mathcal{F}(\mathcal{D}, \theta) \approx \sum_{i=1}^{n} \log q(\boldsymbol{w}^{(i)}|\theta) - \log P(\boldsymbol{w}^{(i)}) - \log P(\mathcal{D}|\boldsymbol{w}^{(i)}) \tag{2.8}$$

an unbiased Monte-Carlo estimator of ELBO. Here $\boldsymbol{w}^{(i)}$ denotes the i:th monte-carlo sample drawn the the posterior. The proposition 2.7 follows from the reparametrization trick.

From the terms in eq. 2.8:

1. $\log q(\boldsymbol{w}^{(i)}|\theta)$ is the log likelihood of weights given the current posterior distribution.

2. $-\log P(\boldsymbol{w}^{(i)})$ is the negative log likelihood of the weights given the prior, which is usually not learned and serves as a regularization mechanism.

3. $-\log P(\mathcal{D}|\boldsymbol{w}^{(i)})$ is the likelihood term, dependent on the data $\mathcal{D}$

The first two terms are grouped together as the complexity cost, while the last term is often called the Likelihood cost. With all of the conditions in place, optimization takes place in pretty much the same way as for basic backpropagation, only updating two variables instead of one (μ and ρ instead of point estimates). One step of the optimization loop is

1. Sample $\epsilon \sim \mathcal{N}(0, 1)$
2. Let $\boldsymbol{w} = \mu + \log(1 + \exp(\rho)) \circ \epsilon$, $\theta = (\mu, \rho)$
   $\circ$ referring to point-wise multiplication.
3. Let $f(\boldsymbol{w}, \theta) = \log(q(\boldsymbol{w}|\rho)) - \log(P(\boldsymbol{w})P(\mathcal{D}|w))$
4. Calculate gradients w.r.t. μ and σ:
   (a) $\Delta_\mu = \frac{\partial f(\boldsymbol{w},\theta)}{\partial \boldsymbol{w}} + \frac{\partial f(\boldsymbol{w},\theta)}{\partial \mu}$
   (b) $\Delta_\rho = \frac{\partial f(\boldsymbol{w},\theta)}{\partial \boldsymbol{w}} \frac{\epsilon}{1+\exp(-\rho)} + \frac{\partial f(\boldsymbol{w},\theta)}{\partial \rho}$

   Then parameters are simply updated as:

   (a) $\mu \leftarrow \mu - \alpha\Delta_\mu$
   (b) $\rho \leftarrow \rho - \alpha\Delta_\rho$

Complexity cost weighting and priors

The cost to be minimizing can be again rewritten as

$$\mathcal{F}_i^\pi(\mathcal{D}, \theta) = \pi_i \mathrm{KL}[q(\boldsymbol{w}|\theta)||P(\boldsymbol{w})] - \mathbb{E}_{q(\boldsymbol{w}|\theta)}[\log P(\mathcal{D}_i|\boldsymbol{w})] \tag{2.9}$$

Here π_i is the relative weighting of the complexity cost in the i:th mini-batch. There are many ways to weigh the complexity cost against the likelihood cost, but an usual constraint is that $\pi \in [0, 1]^M$ and $\sum_{i=1}^M \pi_i = 1$. Graves et al. (2011) [23] used equal weighting as $\pi_i = 1/M$, but Blundell et al. (2015) [11] found the scheme $\pi_i = \frac{2^{M-i}}{2^M-1}$ to offer better performance in their experiments.

The prior distribution $P(\boldsymbol{w})$ is often chosen to be a diagonal Gaussian distribution, because it allows calculating the KL-divergence with regards to the Gaussian posterior analytically. A Gaussian prior placed on weights would be equivalent to L2-regularization, also known as weight decay. Recently, despite of the inability to analytically calculate KL-divergences using it, Gaussian scale mixture priors have also started to be used [11, 50] because

of their properties facilitating optimization. These scale mixture priors, if containing two scales are defined as

$$P(\boldsymbol{w}) = \prod_j \alpha \mathcal{N}(\boldsymbol{w}_j|0, \sigma_1^2) + (1-\alpha)\mathcal{N}(\boldsymbol{w}_j|0, \sigma_2^2) \tag{2.10}$$

where $\sigma_1 > \sigma_2$ and often $\sigma_2 \ll 1$, α is the relative weights of the two scales, and $\boldsymbol{w}_j$ is the j:th weight of the neural network.

Alternative reparametrizations

One problem with conventional Bayes-By-Backprop is that the same ϵ is used for each parameter in a layer. One side effect of this is that gradients of different weights are correlated, hindering the training.

Kingma et al. (2015) [27] introduced a modification to the reparametrization trick, called the *local reparametrization trick* (LRT). This modification works similarly but rather than weights, pre-activation layer outputs are sampled using ϵ, with a different ϵ for each output. This reduces variances in a minibatch, allowing for faster training overall. Kingma et al. showed that the unbiased Monte-Carlo estimator of log likelihood in Blundell et al. has variance that does not decrease with batch size because of the contribution of batch member covariances due to shared ϵ, and consequently designs an estimator with zero covariance, leading to the method above. Gradient variances with LRT thus are inversely proportional to batch size, leading to easier optimization with higher batch sizes.

Flipout by Wen et al. [54] is another modification to Bayes-by-Backprop, attempting to solve the same problem. While LRT can only be used for fully-connected feed-forward neural networks with no weight sharing, Flipout can be used on other architectures too [54]. Flipout works the same as BBB, except for the fact that it multiplies the sampled ϵ by a random sign matrix of the size of the parameter space. This way the ϵ are to some degree made pseudo-random, decreasing the gradient variances for a very meager computational cost.

2.3.3 Predictive uncertainty estimation and decomposition

Predictive uncertainty is defined as the total uncertainty arising in practice when making predictions with a probabilistic or ensemble based model. Broadly speaking, in deep learning predictive uncertainty can be divided into two main categories. These are *aleatoric uncertainty*, which originates from

the input data, and *epistemic uncertainty*, which originates in the inaccuracy of the model. Epistemic uncertainty can be reduced with more training or some other changes, while aleatoric uncertainty can not be by definition. Furthermore, aleatoric uncertainty is classified into homoscedastic and heteroscedastic aleatoric uncertainties in the context where one tries to estimate it. The former assumes that all observation share the same underlying uncertainty, while the latter allows each observation uncertainty to differ. [50]

Characterization of uncertainty is important because not taking it into account might lead to over-estimation or under-estimation of failure probability of a model, as explained by Der Kiureghian [28]. There exists techniques to decompose predictive uncertainty into aleatoric and epistemic uncertainties in deep learning. To model the epistemic component, these techniques base themselves on ensembles just like those produced by Bayesian Neural Networks, or the variability between ensemble members to be more precise. Strategies for modeling the aleatoric part differs between classification and regression tasks. In classification, heteroscedastic aleatoric uncertainty can be directly inferred from class logits without any network modification [29, 50]. For regression on the other hand, there is a need to encode the uncertainty in the data as it is not there in the first place. By modeling the likelihood as homoscedastic Gaussian likelihood, one can learn a common homoscedastic uncertainty term for the dataset. By modeling the data as heteroscedastic Gaussian likelihood on the other hand, one can learn a different uncertainty term for each data point. This is done by separating the output of the network and its aleatoric uncertainty into two different channels towards the end of the network, and plugging these in the Gaussian likelihood accordingly [26].

Uncertainty in Radar-based Nowcasting

In radar-based nowcasting, the aleatoric uncertainty of models can be divided into that emanating from measurements and that coming from other factors induced by the processing of data inside the nowcasting process. Measurement uncertainty can again be subdivided into sampling-based and system-based uncertainties. Sampling uncertainty refers to the randomness coming from which hydrometeor or obstacle the radar beam encountered in the scanning process, and system-related ones come from various factors in the radar itself. According to Cao et al. [14], sampling-based uncertainties dominate in well-configured radar systems. Other factors inducing aleatoric uncertainty include all stages where information is possibly lost, such as data compression operations. Additionally, for methods where model outputs are iteratively fed back into the algorithm to produce new predictions, the whole compound

uncertainty of the previous step output is accounted as aleatoric uncertainty in the next step.

Prediction error and uncertainty are related to each others but not equivalent. Sources of errors can be used to partly infer sources of uncertainty, keeping in mind that model outputs may have high bias (possible error) but low variance (uncertainty). Bowler et al. enumerates sources of prediction errors in the context of advection based models, dividing them into three categories. These are errors in estimating the initial advection field, in modeling its time evolution and the Lagrangian evolution of features [12]. Part of these errors are related to the model itself, and as epistemic uncertainty can by definition be reduced, these sources of error can be in decreased in practice. This can be achieved by for example improving data quantity and quality with regards to the task at hand, improving the functional model to better express dependencies between outputs and outputs, and by improving the training procedure by choosing more appropriate inductive biases, such as a more fitting prior in Bayesian Deep Learning.

Chapter 3

Materials and Methods

From now on are presented the data and models used for experiments, and how those nowcasting models were verified.

3.1 Dataset and data selection

As input data we use lowest elevation angle radar reflectivity composites with 1km spatial resolution and 5 minute temporal resolution from the Finnish Meteorological Institute, cropped into a 512x512 km region covering southern Finland. The domain and bounding box are illustrated in Figure 3.1.

Because lowest elevation angle horizontal reflectivity is a good estimator of precipitation at ground level, it is a natural data choice for building a nowcasting model. Radar composites made of these radar scans are readily available at FMI which facilitated their retrieval for this work. Additionally, the bounding box covering southern Finland did not have any major data quality issues standing out as opposed to other candidate datasets, and finally missing pixels were correctly labeled. All of these factors contributed to the choice of the data source.

The dataset was chosen so that at first, a selection of rainy days were chosen as to correspond to the 100 days with the most pixels exceeding a 35 dBZ reflectivity threshold during the summer period spanning from May to September during years 2019, 2020, and 2021.

This dataset is then cleaned and filtered, after which it is divided into training, validation, and test sets. Cleaning the data involves first going through all timestamps and removing those with either partially or completely missing data. The filtering part consists of removing timestamps with less than 1% of pixels containing reflectivity values exceeding 20 dBZ. Splitting of data into training, validation, and testing sets is performed by

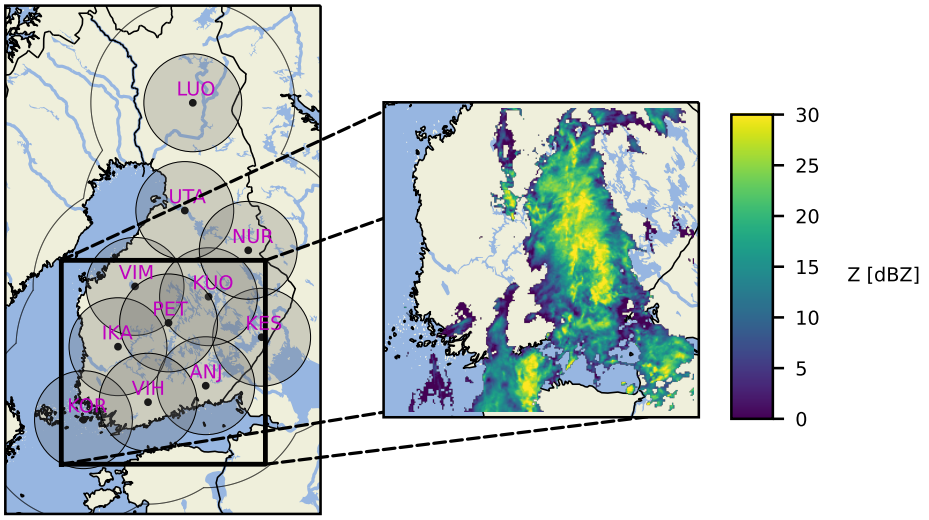


Figure 3.1: FMI radar domain and the chosen bounding box covering southern Finland. The three letter codes identify the radars, inner circles their worst case effective range (usu. during summer), and outer circles their best case effective range (usu. during winter). The lowest level reflectivity composite from the 30th of June 2020 at 00:05:00 is depicted on the right.

using a block sampling strategy [46] with 6 hour long blocks to prevent autocorrelation between consecutive radar images from invalidating independence between splits. The final split sizes were 15840 radar images for the training split, 2664 for the validation split, and 2448 for the test split.

Radar composites are stored as gzip-compressed PGM composite files holding uint8 values, which are converted to reflectivity (dBZ) by applying the relation $\text{pixel\_value}_{\text{dBZ}} = -32 + 0.5 * \text{pixel\_value}_{\text{uint8}}$. When fed into neural networks, the composites are read from HDF5 files as this improves reading speed on distributed storage systems.

From radar reflectivities, rainrate estimates were calculated using Eq. 2.1 solved for R, with empirically determined parameters $A = 223$ and $b = 1.53$, and $z = 10^{Z/10}$, giving us a relation of

$$R = (10^{Z/10}/223)^{1/1.53} \tag{3.1}$$

for the current data.

3.2 Model

3.2.1 The baseline: RainNet

For the implementation of a Bayesian Convolutional Network, we use as our base functional model RainNet [7], which is itself heavily inspired by U-Net and SegNet model families. RainNet just like those families adopts a U-shaped encoder-decoder architecture. The encoder is composed of successive pooling and convolutional layers, reducing image sizes passed to the next layer while increasing the number of filters. The decoder part adopts a mirror archictecture of successive upsampling and convolutional layers, increasing the image size while reducing the number of filters. There is 5 levels in the encoder-decoder structure, just like in SegNet. Additionally, there are skip connections from the encoder to the decoder branch, carrying higher-level filter maps through, just like in U-Net. This is done in order not to lose smaller spatial scale details with pooling.

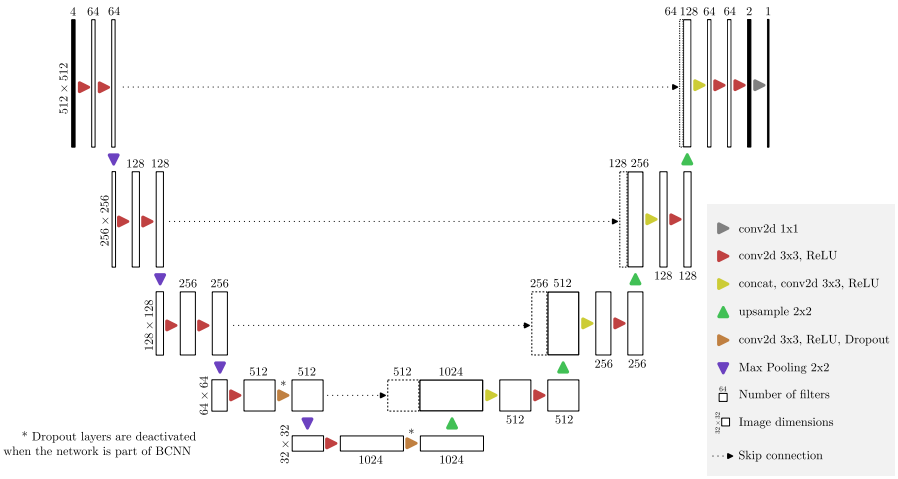


Figure 3.2: The functional CNN architecture used for this work, Identical to that of RainNet from Ayzel et al. [7]. `concat` refers to the concatenation of feature maps along the channel dimension.

The convolutional filters have a filter size of 3, with padding designed to preserve image shape, and a stride of one. Finally, the last convolutional layer in the network has a filter size of 1 with no padding, and it is followed

by a linear activation output layer, i.e. no (nonlinear) activation function. The network does not contain any fully-connected layers, and in total consists of 31.4M parameters. RainNet works by feeding in 4 consecutive radar images, with the physically speaking temporal dimension represented in the channel dimension of the tensor. As the output of the network, we get one output radar image, representing the next predicted frame. In order to make predictions at further leadtimes, the predictions are fed back to the inputs one-by-one in an iterative process.

The network is trained by calculating a loss function between observed and predicted radar images. This loss function is originally the LogCosh loss, as described in Ayzel et al.'s paper [7]. In addition to that loss, the use of Multi-Scale Structural Similarity Index (MS-SSIM) [53] as a loss function was also implemented in this work. MS-SSIM is a loss function, originally an

The rainrates were scaled using a logarithmic transformation $x_{\log} = \ln(x+0.01)$ For the use of LogCosh loss, and they were additionally shifted upwards by a factor of 5 and then scaled down by a factor of 10 to fit into the range of 0 to 1 when using MS-SSIM loss. One benefit of using MS-SSIM instead of Logcosh loss is that MS-SSIM better preserves structural details of predictions such as edges and gradients, which helps with nowcasting higher reflectivity values. This same behaviour however often produces diverging nowcasts, characterized by loss of total precipitation mass and unphysically high precipitation intensities at further leadtimes.

This diverging behavior is alleviated by calculating the loss function for nowcasts done over several leadtimes, making the resulting learned network more temporally stable. Hence for training, the predictions are calculated for 6 leadtimes (30 minutes ahead) and the loss is calculated such that each prediction leadtime is weighted equally. Using multiple leadtimes is also implemented for the LogCosh loss function. MS-SSIM was calculated using a data range parameter of 1.0 and equal scale weights of $[0.2, 0.2, 0.2, 0.2, 0.2]$ ordered from smallest to biggest, as opposite to the scaling optimized for visual perception introduced by Wang et al. [53] in the context of using MS-SSIM as a visual quality assessment tool.

The reason for adopting RainNet as the functional architecture of the Bayesian Neural Network is that compared to for example ConvLSTM based models, it has faster inference without losing much in skill. This is important when building upon a model, especially when having limited resources, as added components make the training and inference slower.

3.2.2 BCNN: a Bayesian extension to RainNet

!FIXME **BCNN is a working name, find potentially better name** FIXME!

BCNN is the Bayesian extension of the RainNet made for this work. In the network, posterior probability distributions of weights and biases are modeled as diagonal Gaussian distributions and optimization is carried out using variants of the Bayes-by-Backprop algorithm described by Blundell et al. (2015) [11], minimizing the ELBO loss function as described in section 2.3.2. Posteriors for all parameters are initialized identically to a mean μ of 0 and a variance σ of 1e-4. Parametrizing the posteriors as Gaussian distributions doubles the number of learnable parameters to 62.8M. The Local Reparametrization Trick (LRT) [27] and Flipout [54] are both used in an attempt to reduce gradient variance and speed-up optimization.

As for the prior distribution, two variants are implemented. One being a diagonal Gaussian prior $P(\boldsymbol{w}) \sim \mathcal{N}(0, 0.1)$ and the second a Gaussian scale mixture prior as defined in Eq. 2.10, with $\alpha = 0.5$ and $\sigma_1, \sigma_2 = 0.3, 0.03$. The scale-mixture prior might enable faster initialization and more accurate asymptotic behavior, but the Gaussian prior has more convenient mathematical properties, as it enables calculating KL-divergences in closed form when combined with a Gaussian posterior as in here, using the analytical expression

$$\begin{aligned} D_{KL}(q(\mathcal{D}|\boldsymbol{w})||P(\boldsymbol{w})) = \frac{1}{2}[\log \frac{|\sigma_P|}{|\sigma_q|} - d \\ +(\mu_q - \mu_P)^T \sigma_P^{-1}(\mu_q - \mu_P) + tr\{\sigma_P^{-1}\sigma_q\}] \end{aligned} \tag{3.2}$$

where q and P denote the multivariate diagonal posterior and prior, μ_q and μ_P their respective means, σ_P and σ_q their respective standard deviations and d the identity matrix of the number of dimensions equal to that of the distributions.

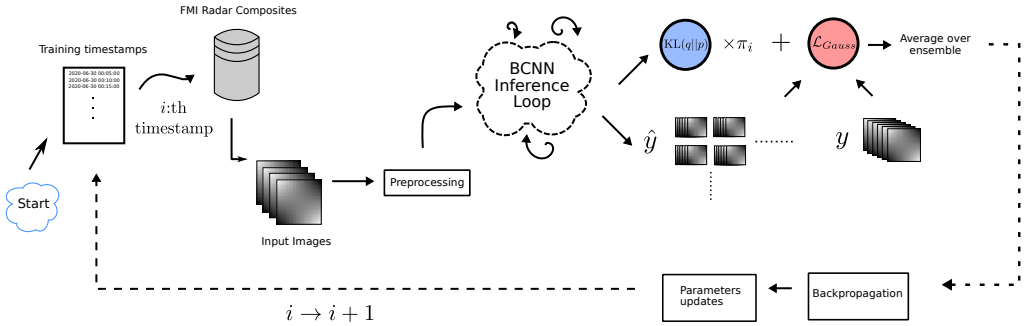
In BCNN, the radar images were scaled between 0 and 1 using the same procedure as described for the MS-SSIM loss function in the context of RainNet in section 3.2.1. The data likelihood cost was modeled as Homoscedastic Gaussian likelihood [26], defined by

$$\mathcal{L}_{Gauss}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2\sigma^2} ||y_i - \hat{y}_i||^2 + \frac{1}{2} \log \sigma^2 \tag{3.3}$$

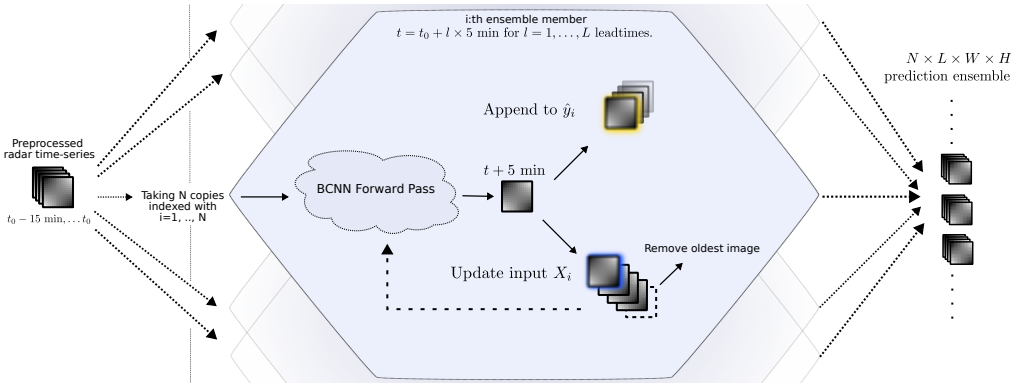
where $i = 1 \dots N$ indexes the number of pixels in the images, and σ here refers to the observational noise parameter. In homoscedastic gaussian

likelihood, aleatoric uncertainty, characterized by σ is assumed to be the same for all the data. This parameter can be learned or left constant. The latter is done in the present work, and a value of 0.02 is chosen as a guess, aiming to represent the uncertainty inherent to radar images.

Multiple procedures for weighting the complexity cost against the likelihood cost are implemented. In addition to the equal weighting scheme from Graves et al. [23] and the batch index dependent scheme from Blundell et al. [11], a scheme reducing the weight of the complexity cost each epoch, defined as $\pi_j = 2^{-j}$, where $j = 0, 1, \ldots$ is the current epoch, is implemented to provide better potential convergence, as the equal weight scheme sometimes provides excessive regularization and the second scheme has problems with converging when using smaller batch sizes as in this work.



(a) Training procedure for the BCNN illustrated. **Rough first version: cleaning up later with radar imgs/preds over black boxes, alignment, sizes, fonts same everywhere after getting feedback on overall structure**



(b) Inference procedure for generating ensemble nowcasts with the BCNN.**Same comments as above...**

Figure 3.3: Training and inference procedures

The predictions are made by simply performing a forward pass through the network, which samples the parameters and produces a Monte-Carlo (MC) sample, i.e. an ensemble member. Predictions for further leadtimes are made in the same way as with the deterministic RainNet, that is by iteratively re-plugging the output, now a MC sample, into the inputs of the network. It is important to note that stochasticity is preserved because parameters are re-sampled at each step of the iterative prediction. Ensembles are produced by repeating this process multiple times independently, that is, such that one initial MC sample will lead to a no more than a single new MC sample each leadtime. Sampling is done this way to avoid the problem of exponential growth that would appear if each sample at leadtime t would yield more than one sample at leadtime $t + 1$. The inference procedure is illustrated in Figure 3.3(b).

Training is performed such as to train the model to predict on a single leadtime (+5min), and after this performing further training on 6 leadtimes (+30min) until model convergence. This is done in an attempt to speed up model convergence compared to training the model on many leadtimes right from the start. At each training step, one MC sample is drawn from the network and gradients are calculated from its ELBO loss. A diagram illustrating the training process is shown in Figure 3.3(a).

3.2.3 Additional Hyper-parameters for NN

Both for RainNet and BCNN, the Adam optimizer is used with an initial learning rate of 1e-04, without any learning rate scheduler. For RainNet, other hyper-parameters were left to their `torch.optim.Adam` default values, while for BCNN, the momentum parameter β_1 was increased to 0.95 to better accommodate the increased stochasticity of SVI compared to non-Bayesian NN [1]. Early stopping was set with condition that the validation loss does not decrease for 3 epochs for both models. For BCNN, only the likelihood part of the loss was taken into account in the early stopping criterion.

3.3 Verification Methods

3.3.1 Evaluation of nowcast predictive uncertainty

In order to visually assess ensemble nowcasts, ensemble mean and standard deviations of nowcasts were plotted and compared to radar observations. In addition, rainrate exceedance probability estimations for the ensemble were plotted for thresholds of 0.5 and 5.0 mm/h. Leadtimes chosen for these

visualizations included 5, 15, 30, and 60 minutes or alternatively 5, 60, 120, and 180 minutes for studying longer less skillful prediction time-scales.

Uncertainties present in the data are coming from a diverse set of sources, including both aleatoric and epistemic uncertainties. In order to accurately represent the data distribution resulting from this compound uncertainty, a large ensemble size is needed. Hence, the ensemble size was chosen to be 24 for preliminary small scale, and 48 for full-scale models. Verification might be even more reliable with bigger ensembles, but this would be at the expense of too much storage space needed for predictions, which would be very inconvenient. Chosen sizes were deemed to be a good compromise regarding this.

3.3.2 Baseline Deterministic Models

In order to verify the skill of BCNN, multiple deterministic models were used to produce nowcasts against which those produced by BCNN are compared. First and foremost, the ensemble averages of BCNN are compared against those of RainNet trained with 6 leadtimes, and both LogCosh and MS-SSIM loss, with the configuration outlined in sections 3.2.1 and 3.2.3. Although loss functions used for training the network are not comparable, it still is interesting to compare the skill between the classical predictions and Bayesian version ensemble means, as underlying functional architectures are the same.

In addition, predictions were calculated with a panoply of non-deep-learning models introduced in section 2.1.1 having different levels of refinement, including Lagrangian persistence (Extrapolation nowcast), S-PROG, and LINDA-D.

Concerning baseline models other than RainNet, before all predictions, the bounding box is applied, input data is converted to rainrate using Eq. 3.1 and thresholded at 0.1 mm/h and possible NaN values are set to zero. After predictions are made, these are thresholded at 0.1 mm/h and converted to back to reflectivities by first using Eq. 2.1 with parameters A and b stated in section 3.1 and then converting Z to dBZ.

3.3.3 Baseline Probabilistic Models

Probabilistic skill verification was performed against STEPS and LINDA-P. The former was chosen as it is a well-established and broadly used model offering good performance for a relatively low inference cost of 1-2 minutes to produce a 3 hours nowcast of 24-48 ensemble members on a modern CPU for a domain of 512x512 pixels [38]. It thus makes sense to want BCNN to perform at least on the same level as STEPS, considering that the time

required for inference with it might not be much lower than that. LINDA-P, also known as the probabilistic variant of LINDA, is computationally more expensive but represents the state-of-the-art in terms of probabilistically predicting localized high-intensity rainfall. Consequently, LINDA-P makes for an interesting benchmark for those cases.

The same preprocessing and postprocessing pipelines as for deterministic cases (section 3.3.2) are used with baseline probabilistic models. Also, the same optical flow and extrapolation parameters are used, and again, Pysteps 1.6.1 default values for methods are used unless stated otherwise.

3.3.4 Deterministic Prediction skill evaluation metrics

Deterministic prediction skill scores were calculated in order to compare the raw predictive skill of ensemble nowcasts to the skill of deterministic models. Such comparison is an important facet of verification as low skill in deterministic scores would even for an otherwise competent model mean that it would benefit from being complemented by a stronger deterministic model. In the case of ensemble nowcasting, Deterministic scores were calculated for ensemble means. Implemented deterministic metrics are divided into four different categories, roughly complementing each others. These are continuous, categorical, and spatial scores, as well as radially-averaged power spectral density.

Continuous scores scores are as their name suggests distance metrics used for the evaluation of continuously valued predictions, i.e. regression tasks. The continuous scores used are Mean Error (ME) and Mean Absolute Error (MAE). ME is defined as

$$\mathrm{ME} = \frac{1}{N} \sum_{i=1}^{N} y_i - \hat{y}_i, \tag{3.4}$$

where we sum over the $i = 1, \ldots, N$ pixels in the radar image, y denotes ground truth and $\hat{y}$ the prediction made. The principal utility of Mean Error is detection whether predictions are biased towards too low or too high rainrates at a certain point in time. On the other hand,

$$\mathrm{MAE} = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|, \tag{3.5}$$

only cares about the magnitude of the errors in predictions by taking the absolute value, so it gives an idea of the prediction skill over the images *on average*. Nevertheless, this is not enough to accurately assess the skill of a

nowcast method, because not all pixels are equally important. Additionally, A poor forecast may have good MAE and vice versa. To illustrate this, a forecast failing to predict localized intense rainfall, but otherwise accurately capturing light rainfall over large areas will usually have a small MAE but will have low operational usefulness.

This limited utility of continuous scores serves as a motivation to introduce categorical scores. These are based on the principle of comparing the presence or absence of a rain event in observations and predictions as defined by having a pixel exceeding a threshold value R_{THR}. The categorical scores are defined by dividing events into four categories, that are true positives (TP), i.e rain events that were correctly predicted, true negatives (TN), i.e. lack of rain event that was correctly predicted, false negatives (FN), i.e. rain that wasn't successfully predicted, and false positives (FP), i.e rain that was erroneously predicted. These categories and their relations are illustrated in Table 3.1. Scores derived from those quantities that were used are probability of detection (POD) defined as

Precipitation is ...	**Observed**	**Not observed**
Predicted	TP (Hits)	FP (False Alarms)
Not predicted	FN (Misses)	TN

Table 3.1: Contingency table of categories used. Alternative nomenclature is shown in parentheses.

$$\mathrm{POD} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{3.6}$$

, which simply tells the probability that an event really occuring is correctly predicted. Next, false alarm rate (FAR) is calculated as

$$\mathrm{FAR} = \frac{\mathrm{FP}}{\mathrm{TP} + \mathrm{FP}} \tag{3.7}$$

which reversely indicates the percentage of positively predicted events not actually happening. Critical success index (CSI), which is defined as

$$\mathrm{CSI} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN} + \mathrm{FP}} \tag{3.8}$$

is computed and aims to generally assess the performance of the forecast by taking the proportion of correct positive event predictions out of critically important cases, that is those excluding true negatives but including both

false alarms (FP) and misses (FN). Lastly, the equitable threat score (ETS) defined as

$$\begin{aligned} \text{ETS} &= \frac{\text{TP} - rnd}{\text{TP} + \text{FN} + \text{FP} - rnd}, \\ \text{where } rnd &= \frac{(\text{TP} + \text{FN})(\text{TP} + \text{FP})}{\text{TP} + \text{FN} + \text{FP} + \text{TN}} \end{aligned} \tag{3.9}$$

was computed. ETS aims to improve CSI assessment of forecast skill, by attempting to estimate the amount of random TP among the prediction using the term rnd, and remove that number of data points from the calculations.

The R_{thr} thresholds chosen for the performing verification are 0.5, 1.0, 5.0, 10.0, 20.0, and 30.0 mm/h. 0.5 mm/h corresponds to very light rainfall and should be easy to predict, while higher thresholds like 20.0 and 30.0 mm/h correspond to very heavy rainfall, which are very difficult to predict even for short leadtimes.

In addition to evaluating prediction skill above rainrate thresholds, it is also important to be able to evaluate the nowcast at multiple scales. The reason for this is that bigger scales have more predictability, and so predicting smaller scales is more difficult while also being of high importance in the context of heavy localized rainfall.

As such, spatial verification scores, namely Fraction Skill Score (FSS) and intensity-scale verification using FSS are added to the panoply of metrics used. FSS is a verification metric aiming to estimate the prediction skill above a certain threshold at different spatial scales. It works by calculating a binary threshold exceedance map for forecasts and observations, averaging it over gaussian windows of different lengths representing scales, and calculating for each of those a Mean Squared Error (MSE) skill score relative to a reference low-skill forecast [42]. Calculating spatial verification scores for a matrix of rainrate intensity threshold and spatial scale is known as intensity-scale verification. Such verification was performed using FSS for thresholds of 0.5, 1.0, 5.0, 10.0 and 20.0 and 30.0 mm/h, as well as spatial scales of 1, 2, 4, 8, and 16 km.

Last but not least, the ability to maintain small-scale details as the forecast leadtime increases is related to the skill of the nowcast with regards to the spatial scale. Failure in maintaining those details will results in low skill at small scales and a blurred forecast demonstrated by a dip in nowcast small-frequency power spectral density. Hence the last deterministic verification metric chosen is Radially-Averaged Power Spectral Density (RAPSD). This metric as its name suggests provides a way of calculating power spectral densities for 2D images such as nowcasts, independently of direction.

RAPSD characterizes the amount and type of blurring occurring in deep-learning based models verified. It is calculated for 15, 30, 60, and 180 minute leadtimes.

!FIXME **fix leadtimes, threshs, scales, before submit** FIXME!

3.3.5 Probabilistic Prediction skill evaluation metrics

Deterministic verification scores are not enough to accurately assess ensemble forecast skill, because the advantage brought by ensembles does not lie in their mean value, but rather in the breath and quality of their distributions, allowing to weight in multiple possible scenarios. Consequently, specialized metrics designed to assess probabilistic forecasts are needed. For this work, four different probabilistic metrics are used for verification. These are the Continuous Ranked Probability Score (CRPS), rank histograms, reliability diagrams, and Receiver operating characteristic (ROC) curves.

CRPS is a metric generalizing MAE for probabilistic forecast. It is defined as

$$\mathrm{CRPS}(F, x) = \int_{-\infty}^{\infty} (F(y) - \mathbb{1}(x \geq y))^2 dy \tag{3.10}$$

where F is the forecast cumulative density function (CDF) and $\mathbb{1}(x \geq y)$ is the empirical CDF of the observation x. CRPS aims thus to represent the distance between those cumulative distributions as a proxy to estimate ensemble forecast skill.

A rank histogram is a measure that from the rank of the radar observation among ensemble members, builds a histogram. For each pixel, the rank is determined and and a bin is incremented accordingly. The shape of the histogram is indicative of whether the spread of the ensemble is representative of the true spread of observations. Variability in the ensemble equal to the uncertainty of observations would make a flat histogram. A convex histogram would mean that ensemble spread under-estimates true uncertainty, whereas a concave histogram would mean that uncertainty is over-estimated, that is that the ensemble is more uncertain that it should be. Furthermore, skewness of the histogram gives a hint on whether there is any kind of bias in the predictions of the ensemble, although ME tells the same thing.

ROC curves are a verification method assessing the ability of a forecast to discern between positive and negative events, that is in the present case pixels with or without exceedance of a particular rainrate threshold. This is accomplished by keeping track of false alarm rates (FAR) against probability of detection (POD) of an event. Probabilities are divided into a certain number of bins over which corresponding FAR are averaged, and a curve is

formed. A random forecast corresponding to no skill corresponds to a line, and an increasing area under the curve indicates better discernment ability.

A reliability diagram presents observed relative frequencies of events (rain-rates exceeding a certain threshold value) against their forecast probabilities. When these two values are close to each others, the forecast is said to be reliable. This means that a skillful forecast is defined by being close to the line of equality to observed relative frequencies for all forecast probabilities output by the model.

ROC curves are conditioned on observation of an event, while reliability diagrams are conditioned on its forecast. Because of this they complement each others well in the evaluation of ensemble prediction skill. For probabilistic metrics, the same rain intensity thresholds as in deterministic metrics, namely 0.5, 1.0, 5.0, 10.0, 20.0, and 30.0 mm/h are used. CRPS is calculated for the whole 36 leadtimes covering 3 hours, while other metrics are calculated for leadtimes of 15, 30, 60, 120, and 180 minutes.

!FIXME **Fix LTs, thresholds at the end, after experiments are finished** FIXME!

3.3.6 Practical points regarding verification experiments

All of the scores described in the previous sections complement each others, and forecast skill is thus estimated as a combination of them, as no score is able to capture all of the facets of a great nowcast.

In practice, predictions are calculated for all models for all the timestamps contained in the test set described in section 3.1. Predictions are computed for 36 timesteps, which is equivalent to 3 hours into the future with an interval of 5 minutes. After that, verification metrics are computed using the predictions of each model and they are averaged over the set for each leadtime of interest.

In order to make sure that results are valid, all timestamps having any (even a single) observation missing are discarded, and similarly timestamps having any prediction from any model missing are also removed from calculations. Additionally, only pixels where data is present in the predictions of all models are counted in metric calculations. This is accomplished by calculating a common NaN mask using the logical OR operation over NaN values of each model, and subsequently applying that common mask to each prediction.

Intermediate results are saved such that predictions are saved in HDF5 archives, in a format where each predicted radar image is a separate dataset in deterministic models, and each ensemble of predicted radar images for a certain leadtime is its separate dataset in ensemble nowcast models. In this

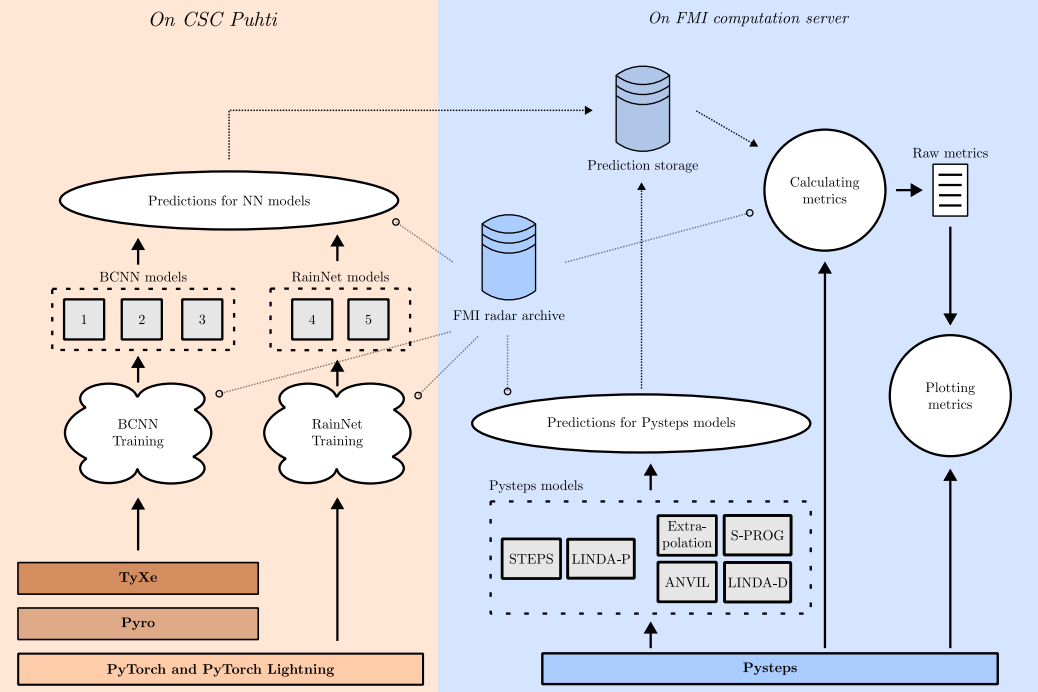


Figure 3.4: Overall workflow used for this work, tied with used libraries for different components.

storage procedure, predicted float reflectivity values are first compressed into UINT8 with a scale-offset scheme, then thresholded to 0.1 mm/h to facilitate further lossless compression which is finally carried out using GZIP. Trying to reduce the storage space needed is essential because predictions take a very large amount of space on disk. This is a consequence of combining a big number of timestamps, ensemble members and leadtimes for multiple models. Raw metrics do not suffer from the same problem and are saved to disk in a binary Numpy format.

3.4 Software and resources

The deep learning models were all implemented using the PyTorch framework and the PyTorch Lightning wrapper [19]. These libraries were chosen because of the combined ease of prototyping brought by PyTorch Lightning, and the maturity and flexibility of the parent framework PyTorch. RainNet was ported to PyTorch following the original Tensorflow implementation by Ayzel et al. [6].

As for the implementation of the probabilistic inference mechanisms for

Bayesian Neural Networks, choice was made not to implement them by hand, but to rely on the machinery contained in the Probabilistic Programming Language (PPL) Pyro [9], which is itself built on top of PyTorch and includes a fully-featured implementation of Stochastical Variational Inference (SVI). In order to facilitate the implementation task, the TyXe package [41] was used. TyXe is a library designed to provide an interface simplifying the implementation of Bayesian Neural Networks using PyTorch and Pyro. A few of the reasons why TyXe was chosen are that it permits easily turning existing neural networks into BNNs without having to use hard-coded bayesian layers, and dynamically switching on-and-off the local reparametrization trick and flipout in layers. Some problems encountered include components of TyXe having difficulties working together with PyTorch Lightning abstractions.

Verification experiments and non-deep-learning baseline models were ran and implemented using the open-source library Pysteps [38]. It provides implementations for all non-deep-learning models described as well as implementation of verification metric primitives used in this work, and tools for their visualization.

The computational resources from the Finnish IT Center for Science (CSC) were used for GPU intensive tasks such as training and calculating predictions with deep-learning models, and one of FMI's computational servers was used for performing other, more CPU-intensive operations such as predicting with baseline non-deep-learning models and calculating verification metrics. We trained the models on the CSC Puhti supercomputer, using one Nvidia V100 GPU with 32GB of VRAM, 64GB of RAM, and 10 cores from a 2.1 GHz Intel Xeon Gold 6230 CPU. As for the FMI server, it contains 2 Intel Xeon Gold 6138 2.0 GHz CPUs with each 20 cores and 2 threads by core, with 192GB of RAM.

Chapter 4

Results

4.1 Case studies for nowcasts

4.1.1 Case study 1 : Large-scale Stratiform rain event

4.1.2 Case study 2 : Rapidly evolving convective rain event

4.2 Deterministic prediction skill (Metrics)

4.3 Probabilistic prediction skill (Metrics)

4.4 Uncertainty estimation

4.4.1 Uncertainties against leadtime

4.4.2 Uncertainties against rainfall intensity

4.4.3 Epistemic uncertainty against training parameters

Chapter 5

Discussion

5.1 Goodness of results

5.2 Validity of results

Variational approximations tend to underestimate uncertainty of learned distributions. [10, 33]. This can be observed from the rank histograms of BRN as a generally concave histogram shape compared to models based on stochastic perturbations. This means that the effect of variability of ensembles being less than that of observations was most indeed most pronouced in variational models, as one might expect.

- Assumption of data uncertainty constant in HGL is questionable
- also chosen ad-hoc value is questionable and might affect result quality
- 2-downsampling : lifetime of features at 2km timescales vs 1km : use of prob. nowcast to average out smallest levels and improve scores: is it justified with this avg pool?

5.3 What could we learn from uncertainty

5.4 What would have to be improved, potential problems in the study?

5.5 Directions for further work

Decomposition of predictive uncertainty into aleatoric and epistemic uncertainties is not performed, as it was deemed out of scope for this work. Indeed, trying to model the uncertainty the radar image inputs is a separate problem which is not so much of interest in the domain of radar-based precipitation, as so much more can be done to improve performance by ameliorating radar scan fidelity, preprocessing, and choice. Despite of this, one could still view aleatoric uncertainty as a proxy to non-predictability of precipitation given starting conditions represented by the input frames. This could in the case of the iterative models presented at least serve as a metric to quantify the uncertainty created by feeding back in previous model outputs and could potentially allow the compound aleatoric and epistemic ensemble spread to better match the true distribution of the data.

Chapter 6

Conclusions

Chapter 7

references

[1] SVI Part IV: Tips and Tricks - Pyro Tutorials 1.8.1 documentation.

[2] Next Generation Weather Radar (NEXRAD), Sept. 2020.

[3] Agrawal, S., Barrington, L., Bromberg, C., Burge, J., Gazen, C., and Hickey, J. Machine Learning for Precipitation Nowcasting from Radar Images, Dec. 2019. arXiv:1912.12132 [cs, stat].

[4] Alexander, C., Dowell, D. C., Hu, M., Olson, J., Smirnova, T., Ladwig, T., Weygandt, S., Kenyon, J. S., James, E., Lin, H., et al. Rapid refresh (rap) and high resolution rapid refresh (hrrr) model development. In *100th American Meteorological Society Annual Meeting* (2020), AMS.

[5] Andersson, T., and Ivarsson, K.-I. A model for probability nowcasts of accumulated precipitation using radar. *Journal of Applied Meteorology and Climatology 30*, 1 (1991), 135–141.

[6] Ayzel, G. Rainnet: a convolutional neural network for radar-based precipitation nowcasting. `https://github.com/hydrogo/rainnet`, 2020.

[7] Ayzel, G., Scheffer, T., and Heistermann, M. RainNet v1.0: a convolutional neural network for radar-based precipitation nowcasting. 21.

[8] Bauer, P., Thorpe, A., and Brunet, G. The quiet revolution of numerical weather prediction. *Nature 525*, 7567 (Sept. 2015), 47–55.

[9] Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Hors-

FALL, P., AND GOODMAN, N. D. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research* (2018).

[10] BISHOP, C. M., AND NASRABADI, N. M. *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[11] BLUNDELL, C., CORNEBISE, J., KAVUKCUOGLU, K., AND WIERSTRA, D. Weight Uncertainty in Neural Networks. *arXiv:1505.05424 [cs, stat]* (May 2015). arXiv: 1505.05424.

[12] BOWLER, N. E., PIERCE, C. E., AND SEED, A. W. STEPS: A probabilistic precipitation forecasting scheme which merges an extrapolation nowcast with downscaled NWP. *Quarterly Journal of the Royal Meteorological Society 132*, 620 (2006), 2127–2155. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1256/qj.04.100.

[13] BUNTINE, W. Bayesian back-propagation. *Complex systems 5* (1991), 603–643.

[14] CAO, Q., KNIGHT, M., FRECH, M., AND MAMMEN, T. Measurement uncertainty and system assessment of weather radar network in Germany. In *2016 IEEE Radar Conference (RadarConf)* (May 2016), pp. 1–5. ISSN: 2375-5318.

[15] DENKER, J., AND LECUN, Y. Transforming Neural-Net Output Levels to Probability Distributions. In *Advances in Neural Information Processing Systems* (1990), vol. 3, Morgan-Kaufmann.

[16] DENKER, J., SCHWARTZ, D., WITTNER, B., SOLLA, S., HOWARD, R., JACKEL, L., AND HOPFIELD, J. Large automatic learning, rule extraction, and generalization. *Complex systems 1*, 5 (1987), 877–922.

[17] DIXON, M., AND WIENER, G. Titan: Thunderstorm identification, tracking, analysis, and nowcasting a radar-based methodology. *Journal of atmospheric and oceanic technology 10*, 6 (1993), 785–797.

[18] FABRY, F. *Radar Meteorology: Principles and Practice*. Cambridge University Press, Mar. 2018. Google-Books-ID: 0aRwCQAAQBAJ.

[19] FALCON, W., AND THE PYTORCH LIGHTNING TEAM. PyTorch Lightning, 3 2019.

[20] FOX, N. I., AND WIKLE, C. K. A bayesian quantitative precipitation nowcast scheme. *Weather and forecasting 20*, 3 (2005), 264–275.

[21] Gal, Y., and Ghahramani, Z. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of The 33rd International Conference on Machine Learning* (June 2016), PMLR, pp. 1050–1059. ISSN: 1938-7228.

[22] Germann, U., and Zawadzki, I. Scale-dependence of the predictability of precipitation from continental radar images. part i: Description of the methodology. *Monthly Weather Review 130*, 12 (2002), 2859–2873.

[23] Graves, A. Practical Variational Inference for Neural Networks. 9.

[24] Hinton, E. Keeping Neural Networks Simple by Minimizing the Description Length of the Weights. 9.

[25] Ho, J., Kalchbrenner, N., Weissenborn, D., and Salimans, T. Axial Attention in Multidimensional Transformers, Dec. 2019. arXiv:1912.12180 [cs].

[26] Kendall, A., and Gal, Y. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? *arXiv:1703.04977 [cs]* (Oct. 2017). arXiv: 1703.04977.

[27] Kingma, D. P., Salimans, T., and Welling, M. Variational Dropout and the Local Reparameterization Trick. *arXiv:1506.02557 [cs, stat]* (Dec. 2015). arXiv: 1506.02557.

[28] Kiureghian, A. D., and Ditlevsen, O. Aleatory or epistemic? Does it matter? *Structural Safety 31*, 2 (Mar. 2009), 105–112.

[29] Kwon, Y., Won, J.-H., Kim, B. J., and Paik, M. C. Uncertainty quantification using Bayesian neural networks in classification: Application to biomedical image segmentation. *Computational Statistics & Data Analysis 142* (Feb. 2020), 106816.

[30] MacKay, D. J. A practical bayesian framework for backpropagation networks. *Neural computation 4*, 3 (1992), 448–472.

[31] Marshall, J., and Palmer, W. The size distribution of raindrops. *Journal of Atmospheric Sciences 5* (1948), 165–166.

[32] McAllister, R., Gal, Y., Kendall, A., Wilk, M. v. d., Shah, A., Cipolla, R., and Weller, A. Concrete Problems for Autonomous Vehicle Safety: Advantages of Bayesian Deep Learning. 4745–4753.

[33] Minka, T. P. A family of algorithms for approximate Bayesian inference. 75.

[34] Oh, J., Guo, X., Lee, H., Lewis, R., and Singh, S. Action-Conditional Video Prediction using Deep Networks in Atari Games, Dec. 2015. arXiv:1507.08750 [cs].

[35] Pan, X., Lu, Y., Zhao, K., Huang, H., Wang, M., and Chen, H. Improving Nowcasting of Convective Development by Incorporating Polarimetric Radar Variables Into a Deep-Learning Model. *Geophysical Research Letters 48*, 21 (2021), e2021GL095302. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2021GL095302.

[36] Prudden, R., Adams, S., Kangin, D., Robinson, N., Ravuri, S., Mohamed, S., and Arribas, A. A review of radar-based nowcasting of precipitation and applicable machine learning techniques. *arXiv:2005.04988 [physics, stat]* (May 2020). arXiv: 2005.04988.

[37] Pulkkinen, S., Chandrasekar, V., and Niemi, T. Lagrangian Integro-Difference Equation Model for Precipitation Nowcasting. *Journal of Atmospheric and Oceanic Technology 38*, 12 (Dec. 2021), 2125–2145. Publisher: American Meteorological Society Section: Journal of Atmospheric and Oceanic Technology.

[38] Pulkkinen, S., Nerini, D., PÃ©rez Hortal, A. A., Velasco-Forero, C., Seed, A., Germann, U., and Foresti, L. Pysteps: an open-source Python library for probabilistic precipitation nowcasting (v1.0). *Geoscientific Model Development 12*, 10 (Oct. 2019), 4185–4219. Publisher: Copernicus GmbH.

[39] Ravuri, S., Lenc, K., Willson, M., Kangin, D., Lam, R., Mirowski, P., Fitzsimons, M., Athanassiadou, M., Kashem, S., Madge, S., Prudden, R., Mandhane, A., Clark, A., Brock, A., Simonyan, K., Hadsell, R., Robinson, N., Clancy, E., Arribas, A., and Mohamed, S. Skilful precipitation nowcasting using deep generative models of radar. *Nature 597*, 7878 (Sept. 2021), 672–677. Number: 7878 Publisher: Nature Publishing Group.

[40] Rinehart, R. E., and Garvey, E. T. Three-dimensional storm motion detection by conventional weather radar. *Nature 273*, 5660 (May 1978), 287–289. Number: 5660 Publisher: Nature Publishing Group.

[41] Ritter, H., and Karaletsos, T. TyXe: Pyro-based Bayesian neural nets for Pytorch. *arXiv preprint arXiv:2110.00276* (2021).

[42] Roberts, N. M., and Lean, H. W. Scale-Selective Verification of Rainfall Accumulations from High-Resolution Forecasts of Convective Events. *Monthly Weather Review 136*, 1 (Jan. 2008), 78–97. Publisher: American Meteorological Society Section: Monthly Weather Review.

[43] Saltikoff, E., Haase, G., Delobbe, L., Gaussiat, N., Martet, M., Idziorek, D., Leijnse, H., NovÃ¡k, P., Lukach, M., and Stephan, K. OPERA the Radar Project. *Atmosphere 10*, 6 (June 2019), 320. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute.

[44] Schmid, F., Wang, Y., and Harou, A. Nowcasting guidelines–a summary. *Bulletin n^0 68* (2019), 2.

[45] Schmid, W., Mecklenburg, S., and Joss, J. Short-term risk forecasts of heavy rainfall. *Water science and technology 45*, 2 (2002), 121–125.

[46] Schultz, M. G., Betancourt, C., Gong, B., Kleinert, F., Langguth, M., Leufen, L. H., Mozaffari, A., and Stadtler, S. Can deep learning beat numerical weather prediction? *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 379*, 2194 (Apr. 2021), 20200097. Publisher: Royal Society.

[47] Seed, A. W., Pierce, C. E., and Norman, K. Formulation and evaluation of a scale decomposition-based stochastic precipitation nowcast scheme. *Water Resources Research 49*, 10 (2013), 6624–6641. _eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1002/wrcr.20536.

[48] Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k., and Woo, W.-c. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting, Sept. 2015. arXiv:1506.04214 [cs] version: 2.

[49] Shi, X., Gao, Z., Lausen, L., Wang, H., Yeung, D.-Y., Wong, W.-k., and Woo, W.-c. Deep Learning for Precipitation Nowcasting: A Benchmark and A New Model, Oct. 2017. arXiv:1706.03458 [cs].

[50] Shridhar, K., Laumann, F., and Liwicki, M. A Comprehensive guide to Bayesian Convolutional Neural Network with Variational Inference. *arXiv:1901.02731 [cs, stat]* (Jan. 2019). arXiv: 1901.02731.

[51] SÃ¸NDERBY, C. K., ESPEHOLT, L., HEEK, J., DEHGHANI, M., OLIVER, A., SALIMANS, T., AGRAWAL, S., HICKEY, J., AND KALCHBRENNER, N. MetNet: A Neural Weather Model for Precipitation Forecasting, Mar. 2020. arXiv:2003.12140 [physics, stat].

[52] TISHBY, LEVIN, AND SOLLA. Consistent inference of probabilities in layered networks: predictions and generalizations. In *International 1989 Joint Conference on Neural Networks* (1989), pp. 403–409 vol.2.

[53] WANG, Z., SIMONCELLI, E., AND BOVIK, A. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003* (Pacific Grove, CA, USA, 2003), IEEE, pp. 1398–1402.

[54] WEN, Y., VICOL, P., BA, J., TRAN, D., AND GROSSE, R. Flipout: Efficient Pseudo-Independent Weight Perturbations on Mini-Batches, Apr. 2018. Number: arXiv:1803.04386 arXiv:1803.04386 [cs, stat].

[55] YIN, J., GAO, Z., AND HAN, W. Application of a Radar Echo Extrapolation-Based Deep Learning Method in Strong Convection Nowcasting. *Earth and Space Science 8*, 8 (2021), e2020EA001621. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2020EA001621.

[56] ZHAO, H., GALLO, O., FROSIO, I., AND KAUTZ, J. Loss Functions for Image Restoration With Neural Networks. *IEEE Transactions on Computational Imaging 3*, 1 (Mar. 2017), 47–57. Conference Name: IEEE Transactions on Computational Imaging.

Appendix A

First appendix

This is the first appendix. You could put some test images or verbose data in an appendix, if there is too much data to fit in the actual text nicely.

For now, the Aalto logo variants are shown in Figure A.1.

(a) In English

(b) Suomeksi

(c) På svenska

Figure A.1: Aalto logo variants