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! — June 27, 2022 — DRAFT!

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

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:**	40
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:**	40
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	Equivalent 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

Contents

Chapter 1

Introduction

!Fixme **Add rest of references, by 22.6, 45min** Fixme!

!Fixme **remove "we" passive** Fixme!

Nowcasting is defined as weather forecasting at very short timescales, from minutes ahead up to several hours. 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, flash flood prevention, and mitigation of economic loss incurred by sudden and heavy precipitation. 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. 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 their computational expensiveness and the low spatial resolution of large-scale NWP, making those systems unfit to predict for example heavy localized rainfall.

!Fixme **add a tiny little bit about det classical nowcast methods as tongue teaser, by 22.6, 45min** Fixme!

To compensate for the shortcomings of NWP in the realm of precipitation nowcasting, many different dedicated models and algorithms have been developed. 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 advection field. Other methods have been developed that track and try to nowcast the evolution of indi-

vidual rain cells, instead of the whole field. 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 convective initiation and life-cycles. 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, their expressive power and the relative cheapness of inference.

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 phenomena we are most interested in are extreme events, and these are the events that we want to prepare for, it makes intuitive sense that accurate and reliable probabilistic nowcasts are primordial for operational decision-making in meteorological crises.

Indeed, the interest in probabilistic nowcasts has grown lately and many new probabilistic models have been introduced in the last decades, notably the Short-Term Ensemble Prediction System (STEPS) [7] and more recently the Lagrangian Integro-Difference equation model with Autoregression (LINDA) [15]. These methods predict ensembles, that are used to estimate underlying distributions of data and precipitation probabilities. They perform reasonably well, but have computationally expensive inference and suffer from the same limitations other advection-extrapolation based methods.

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. [17] 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 we will focus on from now on is instead to model the uncertainty of model parameters rather than that of the data. 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 predic-

tions 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 [12] and autonomous vehicles [13].

In this work, we will approach the problem of making skillful and reliable probabilistic precipitation nowcasts by building an uncertainty-aware Bayesian Neural Networks. A baseline Convolutional Neural Network (CNN) will be turned into a Bayesian Convolutional Network (BCN) with optimization performed using stochastic variational inference (SVI) over model parameters. The model is trained, validated and tested using Finnish Meteorological Institute (FMI) radar reflectivity composites. For 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

!Fixme **Estimate at start: 12-15 pages** Fixme!

2.1 Precipitation Nowcasting

2.1.1 Weather radars and radar products

- physical functioning and paragraph on history of weather radar
- reflectivity products, 2D and 3D
- from reflectivity to RR, limitations of reflectivity
- double-polarization weather radars and Polarimetric products: info on microphysical processes

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

where z is reflectivity, R is rainrate, and A as well as b are empirically determined parameters. Reflectivity is calculated

2.1.2 Overview of weather forecast methods

- Taking a broad look into the NWP process: Data collection, data assimilation, numerical prediction, postprocessing into forecast products
- NWP in the context of precipitation forecasts

2.1.3 Precipitation nowcasting : classical deterministic methods

- Basic principles of advection equations.
- Chronological advancing, with +/- of each method

2.1.4 Precipitation nowcasting : classical probabilistic methods

- approach types, ensemble etc

2.1.5 Machine learning approaches to precipitation nowcasting

-

Multi-Scale Structural Similarity Index (MS-SSIM), originally an image quality assessment metric [22], 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 it or its single-scale counterpart (SSIM)has been shown to reduce blurring in image reconstruction [24] making it a good candidate loss function for nowcasting with neural networks, as it has been seen that these models suffer from excessive blurring, effectively minimizing standard loss functions such as Mean Squared Error (MSE), but failing to predict useful patterns such as heavy localized rainfall. Indeed in recent years, there has been some cases where MS-SSIM or SSIM started have started to be used in

2.2 Bayesian deep learning

2.2.1 Learning probability distributions

- first old studies
- link overfitting to the bias-variance trade-off

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 presented 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.

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 posterior) 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.

2.2.2 Variational inference

Variational inference as a means of training bayesian neural networks was first introduced by Hinton and Camp in 1993 [10]. However it was not used for a long time before breakthroughs by Graves et al. (2011) [9] and Blundell et al. (2015) [6] permitted its use in large-scale neural networks.

- deriving ELBO
- Bayes by Backprop and SVI
- Local reparametrization
- Monte-Carlo dropout as VI
- where else is VI used
- problems with VI

The evidence lower bound (ELBO), also known as variational free energy, is thus defined as

$$\mathcal{F}(\mathcal{D}, \sigma) = \mathrm{KL}[q(\boldsymbol{w}|\sigma)||P(\boldsymbol{w})] - \mathbb{E}_{q(\boldsymbol{w}|\sigma)}[\log(P(\mathcal{D}|\boldsymbol{w}))] \tag{2.2}$$

In order to apply backpropagation, the standard deviation is parametrized 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 allows gradients to flow through the network as sampling operations are external.

- how does variance scale for basic re-parametrization trick?
- more Details on LRT

Kingma et al. (2015) [11] introduced a modification to the reparametrization trick, called the *local reparametrization trick* (LRT). It works similarly but layer activations are sampled using ϵ rather than weights in order to reduce the computational cost, and even more importantly variances in a minibatch, allowing for faster training.

- Flipout and its possible advantages over LRT

Using the reparametrization trick and

$$\frac{\partial}{\partial\sigma}\mathbb{E}_{q(\boldsymbol{w}|\sigma)}[f(\boldsymbol{w},\sigma)] = \mathbb{E}_{q(\epsilon)}[\frac{\partial f(\boldsymbol{w},\sigma)}{\partial \boldsymbol{w}} + \frac{\partial \boldsymbol{w}}{\partial\sigma}\frac{\partial f(\boldsymbol{w},\sigma)}{\partial \boldsymbol{w}}] \tag{2.3}$$

the ELBO cost function (Eq. 2.2) can be rewritten as in a form amendable to minibatch optimization, yielding

- Elaborate on the above proposition by Blundell et al

$$\mathcal{F}(\mathcal{D},\sigma) \approx \sum_{i=1}^{n} \log q(\boldsymbol{w}^{(i)}|\sigma) - \log P(\boldsymbol{w}^{(i)}) - \log P(\mathcal{D}|\boldsymbol{w}^{(i)}) \tag{2.4}$$

Here $\boldsymbol{w}^{(i)}$ denotes the i:th monte-carlo sample drawn the the posterior. From the terms in eq. 2.4:

1. $\log q(\boldsymbol{w}^{(i)}|\sigma)$ 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 also often called the complexity cost.

The cost to be minimizing can be again rewritten as

$$\mathcal{F}_i^{\pi}(\mathcal{D},\sigma) = \pi_i \mathrm{KL}[q(\boldsymbol{w}|\sigma)||P(\boldsymbol{w})] - \mathbb{E}_{q(\boldsymbol{w}|\sigma)}[\log P(\mathcal{D}_i|\boldsymbol{w})] \tag{2.5}$$

Here π_i is the relative weighting of the complexity cost in the i:th minibatch. 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) [9] used equal weighting as $\pi_i = 1/M$, but Blundell et al. (2015) [6] 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 [6, 21] because

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

$$P(\boldsymbol{w}) = \prod_j \alpha \mathcal{N}(\boldsymbol{w}_j|0, \sigma_1^2) + (1-\alpha)\mathcal{N}(\boldsymbol{w}_j|0, \sigma_2^2) \tag{2.6}$$

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.

2.2.3 Predictive uncertainty estimation and decomposition

!Fixme **find good ref on sources** Fixme!

- Where does uncertainty come from
- division into epistemic, aleatoric uncertainty in ML
- Division for classification
- Division for regression

2.3 Related work

!Fixme **Section to include if 2.1.4 and 2.1.5 become convoluted** Fixme!

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

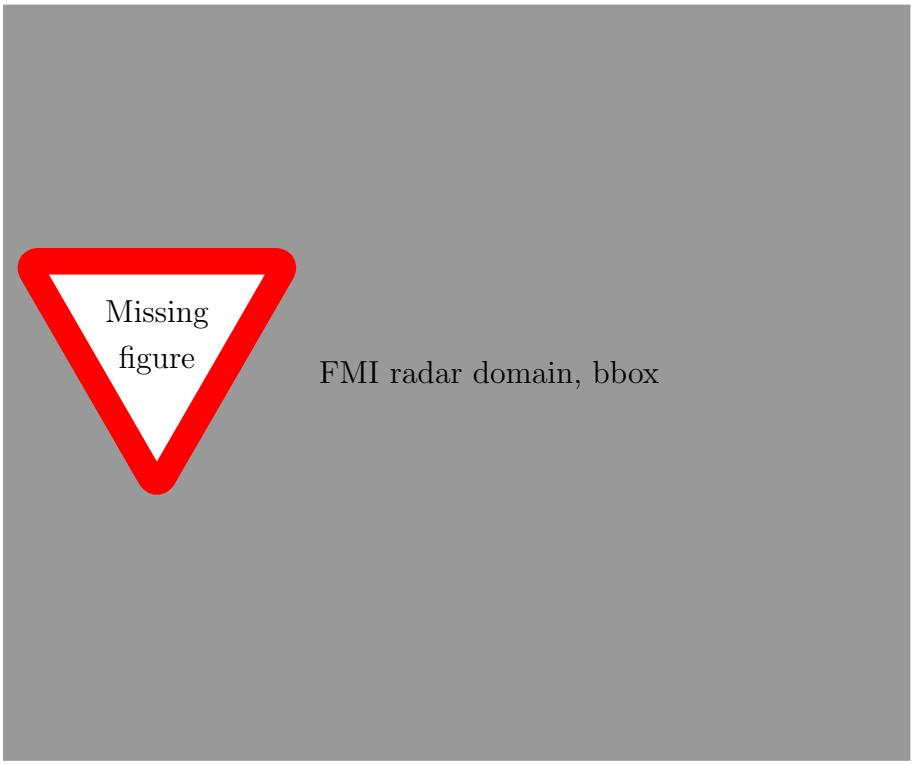


Figure 3.1: FMI radar domain and the chosen bounding box covering southern Finland.

using a block sampling strategy [20] 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 [3], 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.

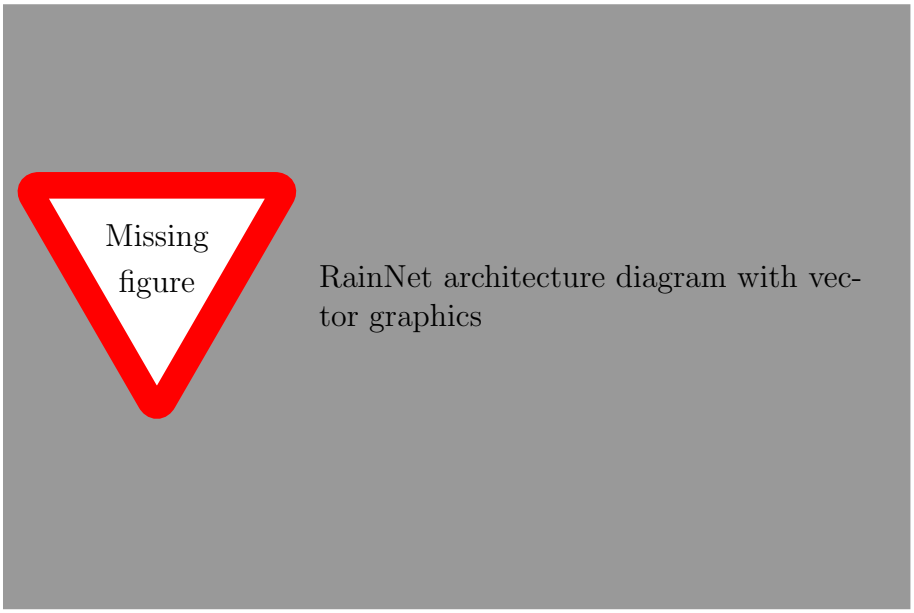


Figure 3.2: The functional CNN architecture used for this work, adapted from RainNet.

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 [3]. In addition to that loss, the use of Multi-Scale Structural Similarity Index (MS-SSIM) [22] 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. [22] in the context of using MS-SSIM as a visual quality assessment tool.

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) [6], minimizing the ELBO loss function as described in section

2.2.2. Posteriors for all parameters are initialized identically to a mean μ of 0 and a variance σ of 1e-4. The Local Reparametrization Trick (LRT) [11] and Flipout [23] are both implemented and attempted for reducing gradient variance.

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. 2.6, 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.

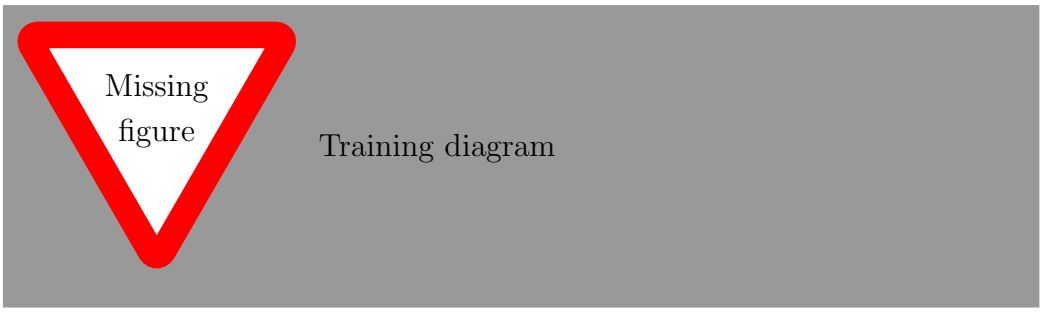
In BCNN, the radar images were scaled between 0 and 1 using the same procedure as described for the MS-SSIM loss function in the context of RainNet in section 3.2.1. The data likelihood cost was modeled as Homoscedastic Gaussian likelihood, defined by

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

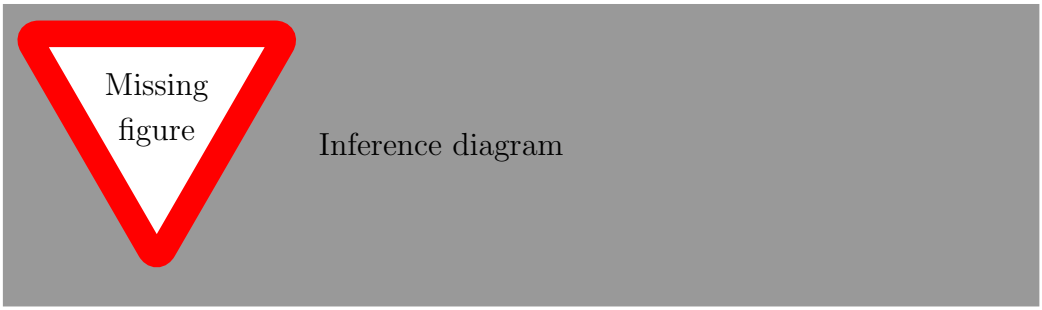
where $i = 1 \dots N$ indexes the number of pixels in the images, and σ here refers to the observational noise parameter. In homoscedastic gaussian likelihood, aleatoric uncertainty, characterized by σ is assumed to be the same for all the data. This parameter can be learned or left constant. The latter 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. [9] and the batch index dependent scheme from Blundell et al. [6], a scheme reducing the weight of the complexity cost each epoch, defined as $\pi_j = 2^{-j}$, where $j = 0, 1, \dots$ is the current epoch, is implemented to provide better potential fitting ability, 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.



(b) Inference procedure for generating ensemble nowcasts with the BCNN.

Figure 3.3: Training and inference processes

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. 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
- LR scheduler (Reduce LR on plateau)

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 2.1.3, including Lagrangian persistence (Extrapolation nowcast), S-PROG, ANVIL, and LINDA-D.

- Lagrangian persistence (advection-extrapolation)
- S-PROG
- ANVIL
- LINDA-D
- RainNet (c.f. above)

3.3.3 Baseline Probabilistic Models

- STEPS
- LINDA-P

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.

!FIXME **small picture showing contingency table to help vizualization** FIXME!

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

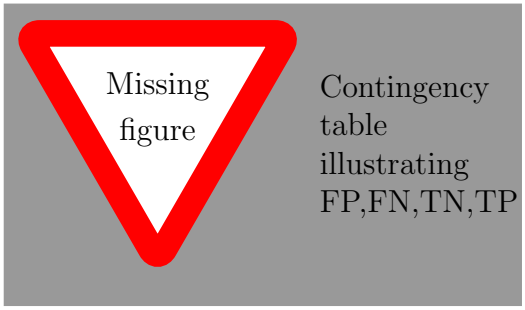


Figure 3.4: Categories used, illustrated using a contingency table.

wasn't successfully predicted, and false positives (FP), i.e rain that was erroneously predicted. These categories and their relations are illustrated in Figure 3.3.4. Scores derived from those quantities that were used are probability of detection (POD) defined as

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

, 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

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

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 [19]. 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 (rainrates 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 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

!Fixme **Some diagram for visualizing grand scheme of workflow** Fixme!

The deep learning models were all implemented using the PyTorch framework and the PyTorch Lightning wrapper [8]. 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

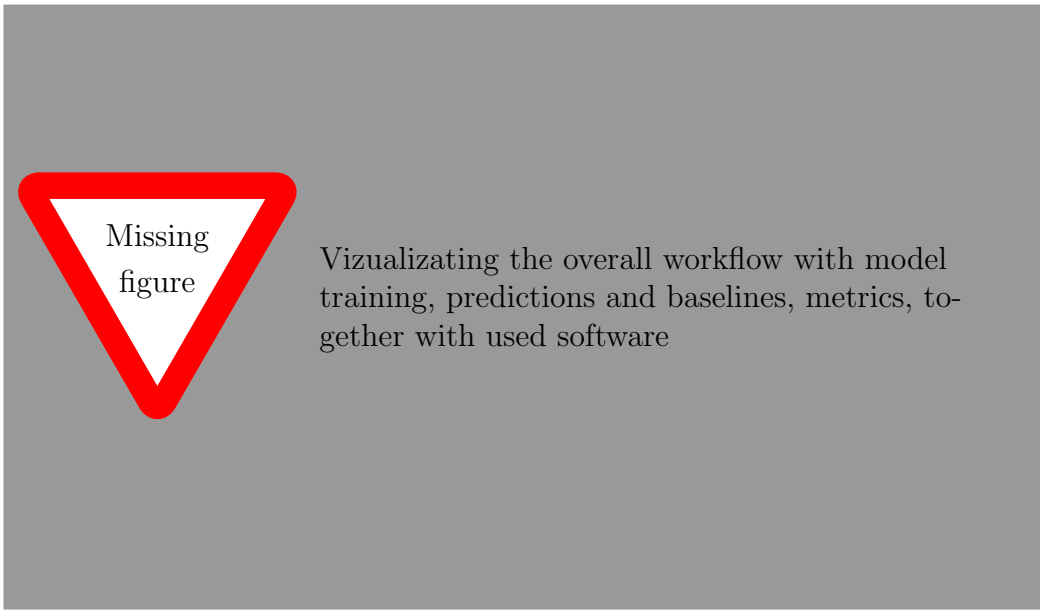


Figure 3.5: Overall workflow used for this work, tied with used libraries for different components.

was ported to PyTorch following the original Tensorflow implementation by Ayzel et al. [2].

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 [4], 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 [18] 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 [16]. 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. [5, 14]. 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

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

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

[4] Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., and Goodman, N. D. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research* (2018).

[5] Bishop, C. M., and Nasrabadi, N. M. *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[6] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight Uncertainty in Neural Networks. *arXiv:1505.05424 [cs, stat]* (May 2015). arXiv: 1505.05424.

[7] 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.

[8] Falcon, W., and The PyTorch Lightning team. PyTorch Lightning, 3 2019.

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

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

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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].

[24] 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