Extreme precipitation nowcasting using deep generative model

Haoran Bi

Delft University of Technology

TU Delft

Extreme precipitation nowcasting using deep generative model

by

Haoran Bi

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on October 14, 2023.

Student number:	5242525	
Project duration:	Feb, 2022 – Oct, 2022	
Thesis committee:	Prof. dr. ir. J. Dauwels,	TU Delft, supervisor, chair
	Dr. J. Yang,	TU Delft

TU Delft

Abstract

Extreme precipitation can often cause serious hazards such as flooding and landslide. Both pose a threat to human lives and lead to substantial economic loss. It is crucial to develop a reliable weather forecasting system that can predict such extreme events to mitigate the effect of heavy precipitation and increase resilience to these hazards.

Numerical Weather Prediction (NWP) models play the dominant role in the field of weather forecasting. However, due to their long computational time, these models had limited utility in predicting weather conditions in the following several hours. This gap is filled by nowcasting, an observation-based method that uses the current state of the atmosphere to forecast future weather conditions for several hours. Operational nowcasting systems typically apply extrapolation algorithms to rainfall radar observations based on simple physics assumptions. However, the physics constraints also limit the performance, and the methods can hardly capture non-linear patterns in the radar observations. Besides the conventional methods, deep learning models have started to play an essential role in this field. Recent works have shown the promising potential of using deep learning models to tackle the nowcasting task, which is also this thesis's focus.

The thesis work mainly studied in two directions: the development of novel deep generative models for precipitation nowcasting and the application of statistical approaches for better modeling and prediction of extreme events. For the first direction, our proposed model is inspired by recently developed deep learning models from the field of visual synthesis. The model makes use of a two-stage structure: the first stage is a Vector Quantization Variational Autoencoder (VQ-VAE) which compresses the original high-resolution radar observations into a low-dimensional latent space. The second stage works in this latent space. It contains an autoregressive Transformer that models the probabilistic distribution of latent space data. The trained Transformer can predict the latent space representation of future frames. ForT better modeling and prediction of extreme events, Extreme Value Loss (EVL) is proposed and incorporated with the autoregressive Transformer. The loss function aims at penalizing predicting extreme cases as non-extreme and predicting non-extreme cases as extreme in order to solve the high imbalance between extreme and normal precipitation data. Our results show that the proposed model shows comparable performance with the state-of-the-art conventional method and other deep learning nowcasting models. The proposed EVL has also been shown to improve the overall performance and accuracy in predicting extreme events.

Preface

In this report I present my master thesis research project, which was conducted between Febuary 2022 and October 2022. This project finalizes my master of Electrical Engineering and mark the end of my student time and also my study at TU Delft. This has been a tough but fruitful journey, and I would like to express my sincere gratitude to everyone who has been a part of this journey.

Foremost, I would like to express my sincere gratitude to my supervisor Prof. Justin Dauwels for the continuous support of my thesis research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. Throughtout the project period, I met many difficulities and the discussion with Justin can always give me insigtful ideas about solving the problems. I would also like to thank the rest of our research group: my daily supervisors Yanbo Wang and Cristian Meo and other two master students Zhiyi Wang and Max Kyriliuk. I'm very grateful having them working as teammates in this project and don't need to face the challenges alone. It has always been pleasant to work with them in this project. Besides, I would also like to thanks Ruben Imohff, who is our external advisor. Thanks for giving us the guidance into the field of nowcasting and the valueable discussion which gives us a lot of inspirations about what we should do to really contribute to this research field. Last but not the least, I would like to thank my family: my parents for supporting me spiritually throughout my life and providing me with the support and freedom to choose my own life. In paticular I'm grateful to my girlfriend Zidong, who supports me all the way and face the difficulities together with me.

Haoran Bi
Delft, October 2022

Nomenclature

Abbreviations

Abbreviation	Definition
ANN	Artifical Neural Network
AUC	Area Under Curve
CE	Cross Entropy
CNN	Convolution Neural Network
CSI	Critical Success Index
EVL	Extreme Value Loss
EVT	Extreme Value Theory
FAR	False Alarm Ratio
FA	False Alarm Rate
FSS	Fractional Skill Score
GAN	Generative Adversarial Network
LR	Learning Rate
LSTM	Long-short term memory
MSE	Mean Square Error
MAE	Mean Absolute Error
NWP	Numerical Weather Prediction
POD	Probability of Detection
PCC	Pearson Correlation Coefficient
QPE	Quantitative Precipitation Estimation
QPF	Quantitative Precipitation Field
RNN	Recurrent Neural Network
ROC	Receiver Operation Characteristic
WCE	Weighted Cross Entropy

Contents

List of Figures

1 Introduction

1.1. Background: Precipitation nowcasting

This thesis work focuses on a kind of weather forecasting called nowcasting. Nowcasting systems aim to predict the future weather condition in the short term (typically less than 6 hours), in which period Numerical Weather Prediction (NWP) systems have limited use. NWP is the most commonly used method for weather forecasting; it makes use of mathematical models of the atmosphere and ocean to predict future weather conditions. However, because of its complexity, in the early years, the system typically had lower resolution and much longer computational time compared with nowcasting systems, making it unsuitable for short-term prediction tasks [23]. In recent years, with the improvement of computational power, NWP can reach a high resolution but still cannot generate as skillful predictions as the nowcasting system. Even though the nowcasting system can only predict weather conditions for the following 1-6 hours, accurate and reliable nowcasting results are essential for the early warning of serious extreme-precipitation-related hazards such as flooding and landslide.

In the field of precipitation nowcasting, weather conditions are usually represented by the radar precipitation fields produced by weather radars. And the systems are based on various radar-extrapolation methods. In general, the inputs of a nowcasting system are precipitation fields of the previous timestamps (usually the past 1 to 3 hours), and the outputs are predictions of future radar precipitation fields. The figure 1.1 shows an example of the nowcasting result produced by PySTEPS.

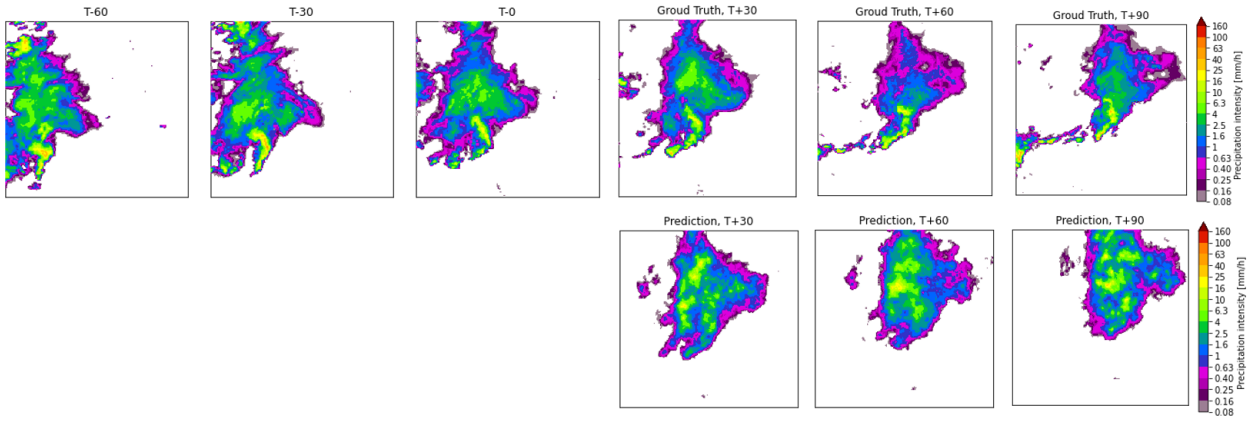

Figure 1.1: An example of the input radar images (T-60, T-30, T) and comparison between the ground truth and predicted radar image (T+30, T+60, T+90) of the nowcasting system (Prediction results produced by PySTEPS)

Weather phenomena can be analyzed at four scales of motion: the global scale, synoptic scale, mesoscale, and microscale (from largest to most minor scale) [23]. At an early age, because of the

limited computational resource, NWP models can only work at low spatial and temporal resolutions, so they can only capture mesoscale patterns but not microscale ones such as small convective patterns, which are essential for predicting rainfall in the near future [23]. Under such circumstance, radar-extrapolation-based methods, which uses the radar observations of the current state of the atmosphere to forecast future weather condition, are proposed. This kind of method has the advantage of a simpler model, higher resolution, and better prediction skills in the early hours (Usually less than 6 hours [6]). Nowadays, radar-extrapolation methods are still the basis of most operational nowcasting systems.

Researchers have also explored the possibility of using deep learning models for nowcasting tasks in recent years. Like the conventional nowcasting methods, most deep learning nowcasting models also use radar precipitation fields to represent weather conditions and try to extrapolate the s future precipitation field. The first deep learning precipitation nowcasting model was proposed by Shi [31] in 2015, called ConvLSTM. This model considers nowcasting as a video prediction task and inventively applies convolution operation in LSTM to capture spatial and temporal features at the same time. Another group of researchers considered it an image transformation task and built the deep learning model based on the U-Net structure [19]. Despite its success in many other deep learning tasks, deep generative models, such as GAN and VAE, have not been widely applied for precipitation nowcasting tasks. However, based on the results in [26], and [16], the deep generative model has shown great potential in producing skillful nowcasting results.

Compared with the traditional radar-extrapolation methods, deep learning models are purely data-driven, flexible, and require no explicit physics constraint. In addition, the use of activation function in deep learning models allows better non-linear modeling capability, which is what conventional methods struggle with and is essential for modeling non-linear weather events such as the initiation of convection [26]. However, the precipitation nowcasting tasks are challenging and different from other well-developed deep learning tasks. Undesirable features like blurry generation are commonly shown in deep learning nowcasting results, and interpretability remains a problem. More detail about these precipitation nowcasting methods is introduced in chapter 2.

1.2. Motivation

Most conventional nowcasting methods are constrained by physics assumption, which often fails to hold in the real world, making the model struggle to capture certain essential patterns. The emergence of deep learning methods in the research field of nowcasting may provide a solution to this problem. These deep learning methods are purely data-driven and can predict future precipitation maps directly without any physical constraints. To this end, multiple deep learning models for nowcasting tasks have been proposed. Although, in general, these models show the potential of producing more accurate predictions for low-intensity rainfall compared with conventional methods, they often have poor performance for heavy or extreme rainy events. In addition, most models tend to produce blurry and unrealistic predictions in a longer lead time [26]. Both of these disadvantages limit their operational utility and motivate this research work.

Nowcasting task is different from traditional deep learning tasks such as video prediction, which is very similar to a nowcasting model in terms of the system input and output. One feature distinguishing nowcasting tasks from these regular tasks is the handling of extreme and out-of-sample events [23]. Such events are considered outliers in most regular tasks and are ignored. However, these events represent rare but heavy rainfall which is crucial in the nowcasting task and may cause substantial damage to the economy and society. These events are not properly handled in most deep learning nowcasting models, and this leads to the loss of extreme precipitation patterns in the prediction. But on the other hand, if we apply classic techniques such as class weight or oversampling to assign a higher weight to the extreme events, this can easily lead to over-fitting problems and overestimate the precipitation [9]. Considering the difficulty in modeling both normal and extreme events, we decided to explore the possibility of building the model with concepts from extreme-value theory.

For the blurry prediction, several researchers [26] have pointed out that it mainly results from the lack of physical constraints and the use of mean square error (MSE). In addition, computer vision researchers

also prove that using adversarial training may lead to sharper and more realistic generations. This theory was also demonstrated in a recent nowcasting research paper by DeepMind [26], where a GAN-based model is proposed for precipitation nowcasting and produces sharp predictions. Considering its promising potential, we also plan to explore different deep generative models for the nowcasting task.

Overall, this thesis work aims at finding and developing better deep learning nowcasting models, which are more effective in predicting extreme events and can produce sharper predictions. In this way, we can move towards an operational deep-learning-based nowcasting system.

1.3. Research objective

This research aims to develop a deep generative model for the nowcasting of extreme precipitation events happening in catchment areas in the Netherlands, with a maximum lead time of 180 minutes and a time interval of 30 minutes. It can be further split into two objectives for our nowcasting system: first, the model is expected to generate skillful nowcasting results for the whole Netherlands. Second, the generated prediction is expected to reliably detect extreme events that happened in catchment areas.

To reach the objectives, this thesis work covers two main topics. The first topic is the development of a novel deep generative model for precipitation nowcasting. The proposed model is inspired by recent research work in visual synthesis, and it makes use of a two-stage structure with a VQGAN in the first stage and an autoregressive transformer in the second stage. The second topic is incorporating extreme-value theory and the deep generative model for better modeling extreme events. The proposed method modifies an extreme-value-theory-based loss function called Extreme Value Loss (EVL) [9] and incorporates it with the autoregressive transformer of our model.

Two research questions can be concluded:

1. How can we develop a deep generative model that can produce a reliable prediction of precipitation field for the following 3 hours?
2. How can we define and detect extreme precipitation events? How can we modify the model correspondingly to further improve the the ability to detect extreme precipitation events?

1.4. Thesis Overview

The thesis is structured as follows:

1. Chapter 2 provides a comprehensive literature review of existing nowcasting methods. These methods include both conventional optical-flow-based nowcasting and typical deep-learning-based methods. PySTEPS, an open-source framework for precipitation nowcasting, is introduced in detail and later applied as the benchmark for our proposed model. Besides the nowcasting methods, extreme value theory, which plays a vital role in this thesis work, is briefly introduced.
2. Chapter 3 focuses on data. It first introduces the KNMI radar dataset used for this project. Two types of radar datasets are compared, and a possible way for conversion between these two datasets is described. The data is then analyzed with analysis based on two levels: pixel-level and catchment-level. Then, based on the data statistics, specific problems are formulated, and the extreme value and extreme events are defined for this thesis project. Finally, the corresponding event selection process is described, and the dataset is formed.
3. Chapter 4 is mainly about methods. Our proposed model and its implementation are described in detail at the start of the section. The model initially applies a cross-entropy loss function, which expects to suffer from the data unbalance problem. So, other options are introduced in this section: the weighted cross entropy loss and extreme value loss (EVL). The EVL, which is inspired by concepts from extreme value theory, is introduced in detail in this chapter. Finally, the verification methods for both the precipitation nowcasting task and extreme event detection task are described.
4. Chapter 5 presents the conducted experiments and their corresponding result evaluation. The experiment is mainly split into two sections. The first section focuses on comparing precipitation nowcasting performance between models trained with different loss functions. In addition, the

effect of other techniques like averaging and post-processing are also studied in this section. The second section focuses on extreme event detection ability. The detection performance is evaluated in two ways: one with defined and fixed extreme thresholds for different catchment areas, the other with the same sets of extreme thresholds for catchments to assess the overall detection performance.

5. Chapter 6 includes the conclusion of the thesis. Specifically, the attempts and results of the proposed research questions are concluded in detail. Besides, the chapter discusses possible future directions of this project, where the short-comes of this thesis project are pointed out, and corresponding recommendations and future suggestions are given.

2

Literature study

The literature study focuses on two research topics: existing nowcasting methods and extreme value theory. The existing nowcasting methods include conventional and deep learning methods, which are compared in the first section. The extreme value theory is a branch of statistics focusing on the probability of extreme events. Its fundamental theories and applications are introduced in the second section.

2.1. Existing nowcasting methods

2.1.1. Conventional methods

Conventional precipitation nowcasting methods can be divided into numerical weather prediction (NWP) based methods and radar-echo-extrapolation-based methods. Given the complexity and long computational time of NWP methods, most operational nowcasting systems are based on radar echo extrapolation algorithms, which is also the focus of this section.

The radar-echo-extrapolation-based methods try to incorporate precipitation-related physics into simple models, such as Euler persistence and Lagrangian persistence [12]. The Euler persistence uses the most recent observation as the prediction. It can be expressed as:

$$\hat{\Psi}\left(t_0+\tau, x\right)=\Psi\left(t_0, x\right) \tag{2.1}$$

Where $\hat{\Psi}$ is the predicted precipitation field, t_0 is the initial time, and τ is the time difference. The Lagrangian persistence assumes the constant state of air parcels, and the change of precipitation field is caused only by background flow and can be expressed as:

$$\hat{\Psi}\left(t_0+\tau, x\right)=\Psi\left(t_0, x-\lambda\right) \tag{2.2}$$

Compared with the Euler persistence expression, the new variable λ is the displacement vector. Most of the radar-based nowcasting methods are based on Lagrangian persistence.

Based on this assumption, optical flow algorithms from computer vision can be used for nowcasting. The algorithms generally contain two significant steps: estimating the motion field from observations and advecting the latest observations along the motion field to generate predictions [23].

Lagrangian persistence has shown to be applicable for precipitation nowcasting and provides the foundation for many nowcasting systems. However, the assumption frequently fails in the actual motion of the precipitation field. To further improve the performance of the nowcasting model, probabilistic and stochastic approaches, which model not only advection but also the uncertainty of predictions, are proposed. These approaches mainly try to relax the Lagrangian persistence, allowing the change in the advection field [23].

The open source project: PySTEPs, which contains a bunch of precipitation nowcasting algorithms mentioned above, is implemented as a benchmark for this project. It will be introduced in detail in the following section.

Benchmark: PySTEPs

PySTEPS is an open-source and community-driven Python framework for precipitation nowcasting [24]. It provides various algorithms for building a nowcasting system and allows both deterministic and probabilistic configurations. The system has been widely used and considered state-of-the-art in nowcasting tasks.

The core algorithms for PySTEPS's deterministic and probabilistic configurations are S-PROG (Spectral PROGnosis) and STEPS (short-term ensemble prediction system), respectively. STEPS is a probabilistic forecasting method that blends the nowcasting result with the down-scaled NWP result. KNMI has recently employed it as the new operational nowcasting system and will also be used as the benchmark for this thesis work. The implemented PySTEPs method in this thesis work has the following configuration (follows the same configuration as [14]):

- Lukas-Kanade optical flow method [22]
- Backward semi-Lagrangian advection method [12]
- STEPS nowcasting method [5]
- Non-parametric noise method
- FFT (Fast Fourier Transform) for spatial composition [29] [28]
- AR (Autoregressive) model of order 2 [28]
- Lead time-dependent masking method
- Cumulative distribution function for probabilistic matching
- 20 ensamble members

Specifically, the workflow of PySTEPS is as follows:

1. Read radar composites, transform the radar reflectivity data to rainfall (mm/h), then log-transform the result to dB scale.
2. Use the optical flow method to determine the motion filed.
3. Use the advection method to extrapolate future radar precipitation field.
4. Use FFT to decompose the rainfall field into a multiplicative cascade, with each level representing a different spatial scale and rainfall lifetime. An example of the decomposition result is shown in figure 2.1
5. Estimate the auto-correlation matrix for each cascade level, then estimate parameters for an AR model using Yule-Walker equations, and apply the model in time to handle temporal evolution and correlation within precipitation structure. The AR model is expressed as the equation below:

$$R_j(x, y, t) = \phi_{j,1} R_j(x, y, t - \Delta t) + \phi_{j,2} R_j(x, y, t - 2\Delta t) \tag{2.3}$$

   where R is the radar map, (x, y, t) is the coordinates, ϕ_j is the model parameter, and j is the number of cascade levels.
6. Add stochastic perturbations to the AR models and advection field. This way, the uncertainty in rainfall intensities and the motion field is considered.
7. Recompose the cascade with the AR model and the stochastic perturbations to get the result of the nowcasting ensemble.

2.1.2. Deep learning based methods

Deep learning methods have recently started to play a key role in precipitation nowcasting tasks. Instead of relying on physics assumptions, deep learning methods are trained with many radar echo maps and show a strong ability to capture complex non-linear spatial and temporal features. Nowcasting

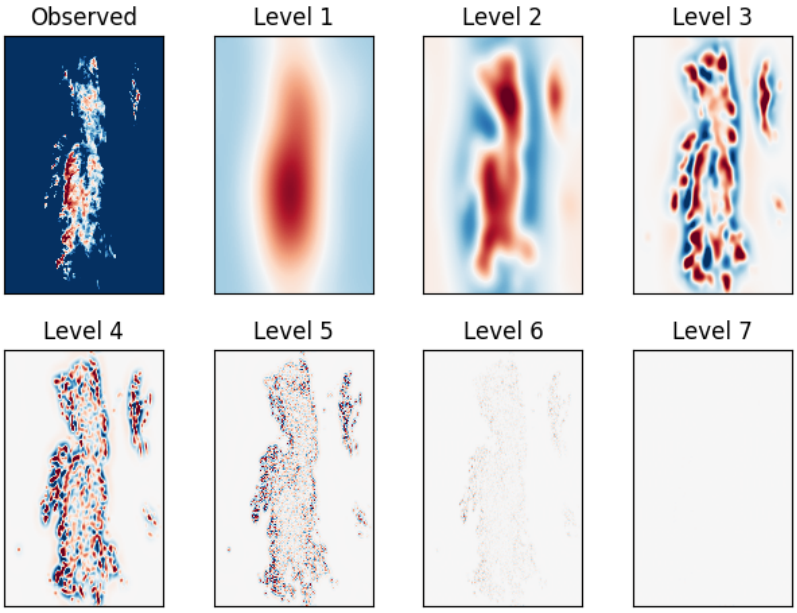

Figure 2.1: An example of different levels of the spatial decomposition [28]

is considered a radar image extrapolation problem in deep learning models like the conventional method. The deep learning models already applied for precipitation nowcasting in the literature can be divided into three main classes: spatial-temporal convolution networks, U-Nets, and deep generative models. These classes are introduced in the following sections.

Spatial-temporal convolution networks

In deep learning, Recurrent Neural Networks (RNNs) are commonly used for temporal modeling and time-series prediction. Its variants, mainly Gated Recurrent Unit (GRU) and Long Short Term Memory (LSTM), make use of the gating mechanism to capture longer-term dependencies and have been widely used in spatial-temporal modeling tasks like video generation and prediction. The first deep-learning-based precipitation nowcasting model: ConvLSTM [31], was proposed based on RNNs. The model replaces the fully connected layer in LSTM's state-to-state and input-to-state transition with convolution operation, enabling better modeling of spatial-temporal features and showing better performance than the previous state-of-the-art conventional precipitation nowcasting method. ConvLSTM is widely used as the basis for deep learning nowcasting model research. For example, this work is further extended to a new model called Trajectory GRU [32], which can learn the location-variants structure. Compared with ConvLSTM, TrajGRU allows the states to be aggregated along learned trajectories. More recently, researchers also proposed using attention and context matching mechanism to improve the long-term performance and the model interpretability. The results show that these two variants achieve better nowcasting performance than the original ConvLSTM in terms of MSE.

Besides the RNN-based models, it is also possible to use a pure convolutional network for this spatiotemporal prediction task. In this case, the dimension of time is handled as a part of convolution architecture. To achieve this, either 2D convolution or 3D convolution can be used. At the current stage, it is still unclear which direction has better performance. Although it has been proven that CNN-based architecture can outperform RNN-based architecture in sequence prediction tasks, this may fail in the case of spatiotemporal prediction.

U-Net and its variants

Instead of using spatial-temporal convolution networks and treating nowcasting as a spatial-temporal prediction task, researchers also proposed to consider nowcasting as an image-to-image translation

problem. In this case, a U-Net can be adopted as the basis for a precipitation nowcasting model. U-Net is a CNN-based encoder-decoder architecture with a “U” shape. The input are observation radar images and it outputs one classification map for the future frame (e.g. each pixel is classified into <0.1mm/h, <1mm/h, <5mm/h or >5mm/h). Since it only outputs one image at a time, the prediction is carried out recursively. Multiple research projects have applied U-Net as the basic structure for their nowcasting model [19][33]. Compared with spatio-temporal networks, this kind of direct prediction allows the model to have a customized forecast period and can lead to better accuracy in the short term. The training process is also more straightforward. However, it is argued that such a recursive generation process would lead to the accumulation of errors in the long term [30].

Deep generative models

The previous deep learning models treat precipitation nowcasting as a deterministic problem, which may not align with the nature of the weather conditions. To output the single best prediction, these models may average over different possible modes, causing blurry issues.

Alternatively, deep generative models are statistical models that learn the probability distribution of data. Then, predictions can be achieved by sampling from these learned distributions conditioned by the current radar observations. Some widely used deep generative networks include GAN and VAE. First proposed in [13], GAN consist of a generator and a discriminator, which are trained alternatively. The generator tries to generate fake samples from the learned distribution, and the discriminator attempts to distinguish the real sample from the generated fake sample. In terms of VAE, it is an encoder-decoder structure and tries to compress the observation into a latent space. The training aim of the VAEs is to minimize the reconstruction loss and the divergence between the encoded latent space and some prior distribution at the same time.

Multiple works have successfully applied GAN or VAE for spatio-temporal prediction tasks. Several deep generative models have been proposed for the precipitation nowcasting task as well. For example, DeepMind proposed their model: Deep Generative Model of Rainfall (DGMR) [26]. Compared with other deep learning models, DGMR generates probabilistic predictions instead of deterministic ones. So, a collection of different predictions can be generated for the same event (Similar to the probabilistic approaches from the conventional nowcasting methods). In terms of performance, it can generate much sharper and more realistic predictions compared with other deep learning nowcasting models. In addition, DGMR also shows generally better performance than the current state-of-the-art conventional method: PySTEPs. Considering the promising potential of deep generative models, it will be the main focus of this project.

Comparisons

Compared with the conventional nowcasting method, the deep learning models’ key advantage is their flexibility. Deep learning models are usually not constrained by any physical assumptions, which may fail in the real world from time to time. Also, their strong non-linear modeling ability allows them to predict non-linear precipitation field patterns, such as the initiation of convection, which are crucial for precipitation nowcasting but may be difficult to predict for conventional nowcasting methods. Despite the great potential, deep learning nowcasting models still have common limitations like blurry prediction, making them not usable for an operational nowcasting system.

Compared between the deep learning nowcasting methods, the U-Net-based and ConvLSTM-based models have two main problems: first, they can have accurate prediction performance for low-intensity rainfall but have poor performance for high-intensity rainfall. Underestimation problems can often happen on the prediction radar map of these models. Second, because of the use of quadratic loss functions such as mean square error (MSE), the models tend to make blurry predictions to compensate for high uncertainty at a longer lead time.

The deep generative model for nowcasting is a more recent research topic than the other two classes. The results of DGMR indicate that it can solve the problem mentioned above. For blurry generations,

using adversarial training can effectively increase the sharpness of the prediction field. In terms of prediction skills, DGMR also performs better in predicting medium-to-high-intensity rainfall.

2.2. Extreme value theory and its applications

Extreme value theory is a branch of statistics dealing with extreme values, which are values with a large deviation from the median of the probabilistic distribution. The theory has been applied in modeling extreme climate such as extreme rainfall [18] and flood [1]. In addition, researchers have also explored the possibility of incorporating with deep learning model. For example, in [9], the theory is incorporated with the loss function; in [4], the theory is Incorporated with a GAN for spatial extremes modeling. The theory deals with both minimal (left tail) and maximal (right tail) cases. However, this section only focuses on the maximal cases since it is more important for precipitation nowcasting tasks.

This section contains two subsections. The first subsection introduces the extreme value distribution, and the second subsection reviews its applications in deep learning tasks.

2.2.1. Extreme value distribution

Extreme value distributions are limiting distributions for the maxima of independent random variables sampled from the same distribution. According to the Extremal Types Theorem (ETT), there are three possible extreme value distributions: Type I, II, and III (also known as Gumbel, Fréchet, and Weibull types). Their corresponding cumulative distribution function (CDF) versions are shown below:

$$\text{I} : F(x) = \exp\left[-e^{-x}\right], \quad -\infty < x < \infty \tag{2.4}$$

$$\text{II} : F(x) = \begin{cases} 0, & x \leq 0, \\ \exp\left(-x^{-\alpha}\right), & x > 0, \alpha > 0, \end{cases} \tag{2.5}$$

$$\text{III} : F(x) = \begin{cases} \exp\left[-(-x)^{\alpha}\right], & x < 0, \alpha > 0 \\ 1, & x \geq 0 \end{cases} \tag{2.6}$$

Type I indicates that the observed data has a light-tailed distribution, such as normal distribution. Type II indicates a heavy-tailed data distribution, often used to model precipitation and economy. Type III indicates a bounded data distribution, which can be used to model variables with a limited range, such as temperature and sea level.

These three distributions are later combined into one distribution called Generalized Extreme Value Distribution (GEVD), which can be expressed as the equation below:

$$H(y) = \exp\left\{-\left[1 + \xi\left(\frac{y-\mu}{\sigma}\right)\right]^{-1/\xi}\right\}, -\infty < \mu, \xi < \infty, \sigma > 0 \tag{2.7}$$

where μ and σ are not the data's mean and standard deviation but are location parameters indicating the center of the distribution and scale parameters indicating the deviation from the center, respectively. The shape of the distribution is governed by the parameter ξ. When $\xi = 0$, the distribution is the same as Type I distribution; When $\xi < 0$, the distribution is the same as Type III distribution; When $\xi > 0$, the distribution is the same as Type II distribution. Figure 2.2 shows an example of the GEV distribution.

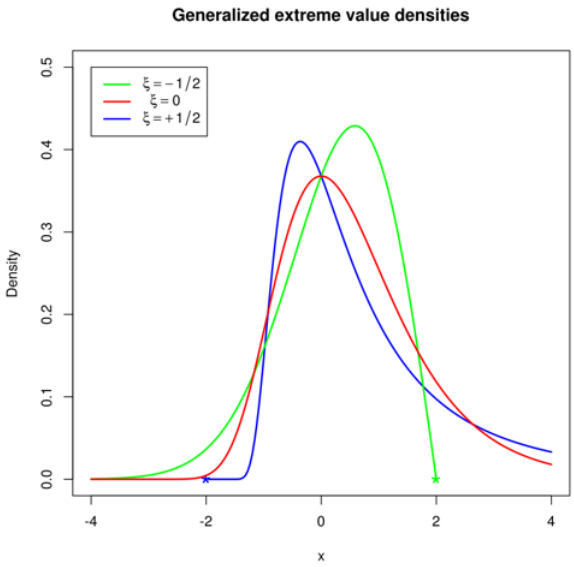

Figure 2.2: Example of GEV distributions with different shape parameter

2.2.2. Applications

Extreme value theory has been incorporated for extreme data analysis in many fields, such as finance, insurance, and hydrology. In this section, we focus on its incorporation with deep learning models to enhance the models' tail distribution modeling ability.

evtGAN

Proposed in [4], evtGAN incorporates the extreme value theory with GANs to model the spatial dependencies between climate extremes (temperature and precipitation). The result of evtGAN indicates that it outperforms both standard GAN and statistical approaches in spatial extremes tasks. Specifically, the evtGAN can be summarized into five steps:

1. Based on the given data (yearly maximum temperature or precipitation), fit the extreme data at each location into a generalized extreme value distribution.
2. Normalize all data to a standard uniform distribution
3. Train the GAN on the normalized data
4. Generate new samples from the trained GAN
5. Normalize the generated data back to the original observation using the previously estimated GEVD

Extreme value loss

Proposed in [9], the author identifies that the deep learning models' weak extreme modeling ability roots in the conventional quadratic loss function and proposed to use incorporate extreme value theory with the loss function to enhance the models' extreme event detection ability in time series prediction tasks.

The model architecture follows a standard GRU structure with two main modifications: the memory network module and the extreme value loss as an additional loss term. In terms of the memory module, the model keeps a record of the previous events' GRU hidden state and their corresponding labels (extreme or non-extreme). Then, for the current prediction, the hidden state can be compared with the memorized states through the attention mechanism and calculated as an extreme indicator (indicates the possibility of current data being extreme data). For the extreme value loss, the author applies a weighted cross entropy term to penalize predicting extreme data as non-extreme and predicting non-extreme data as extreme. The weights are the probability of extreme data and non-extreme data, which can be

estimated from the extreme value theorem.

Extreme value loss is also one main focus of this thesis work. This loss is modified and incorporated with our deep generative model, leading to better extreme detection performance. More details are presented in chapter 4 and chapter 5.

3

Dataset and problem formulation

3.1. Dataset

The section contains three subsections:

- The thesis work focuses on nowcasting for the Netherlands, so precipitation data from The Royal Netherlands Meteorological Institute (KNMI) are used. Two available precipitation dataset, i.e., the RT and MFBS datasets are considered in this thesis and introduced in the first section.
- In the second section, an analysis of the statistics of the RT dataset is presented as it is the main dataset used for this thesis work. Two kinds of analysis are included: pixel level and catchment level analysis.
- The rainfall accumulation converted from the radar reflectivity may not reflect the actual rainfall amount; a possible way to correct this bias in the RT dataset called “CARROTS” [15] is introduced in the third section.

3.1.1. KNMI radar datasets

Meteorologists mainly use weather radar for precipitation observation. In Netherlands, KNMI has two operational C-band weather radars, located at Den Helder and Herwijnen (before 2017, radar station at De Bilt was the operational radar instead of Herwijen) [2]. The location of these KNMI weather radars and a real-time radar map is shown in figure 3.1. The original data produced by weather radars is the

Figure 3.1: Three radar stations in Netherlands: Den Helder, Herwijnen and De Bilt

radar reflectivity Z, which is the amount of back-scattered radiation at 1500m over the Netherlands in the case of the datasets used for this thesis. To estimate the rainfall rate from the radar reflectivity, a fixed Z-R transformation equation [3] can be used:

$$Z_h = 200R^{1.6} \tag{3.1}$$

where Z_h represents the radar reflectivity (unit: mm^6m^{-3}) and R represents the rainfall rate (unit: $mmhr^{-1}$. The original radar reflectivity has the unit of dBZ can be converted to mm^6m^{-3} using $Z_{h(dBZ)} = 10log_{10}(Z_h)$. In this process, reflectivity below 7dB (precipitation intensity < $0.1mm/h$) are ignored and reflectivity above 55dB (precipitation intensity > $100mm/h$) are fixed at 55dB. Also, isolated pixels are ignored. The converted and processed data is then the quantitative precipitation estimation (QPE) data from the KNMI website.

Besides the dataset we described (RT dataset), another dataset is available from the KNMI website. This dataset (MFBS dataset) has the same spatial and temporal resolution as the RT dataset. However, data from a rain gauge network (containing 356 gauges over the Netherlands) are used to adjust the original QPE. In this way, it provides a more accurate rainfall rate estimation and can be considered a reference dataset. However, from the KNMI websites, this dataset updates once a month, and operating the rain gauge network requires lots of manual work, so the data cannot be considered real-time.

A comparison between the raw radar images from these two dataset is shown in the figure 3.2. From the comparison, we can see a large area on the maps is masked for both datasets: the RT dataset only has data within the radars' coverage, while the MFBS dataset only has data available for the land area of the Netherlands. The Delfland catchment area is cropped and compared in 3.3 for better comparison. From the comparison, assuming the MFBS dataset provides the ground truth of the precipitation accumulation, radar images from the RT dataset generally have the correct shape of the rainfall field. However, they tend to underestimate the rainfall intensity heavily.

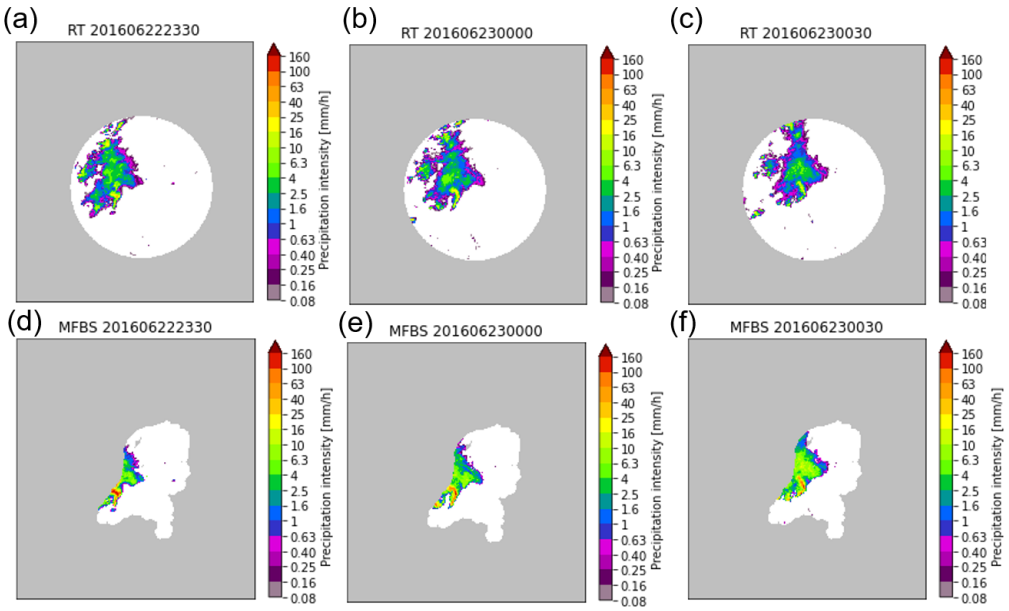

Figure 3.2: Comparison of radar images between RT and MFBS datasets for three different timestamps, (a-c) RT radar images of the selected timestamps, (b-d) MFBS radar images of the selected time stamp (No data in the grey area)

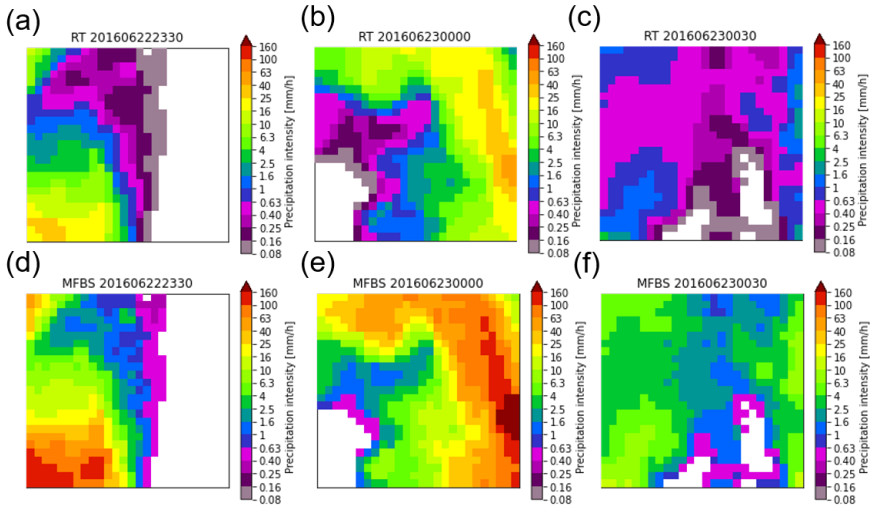

Figure 3.3: Comparison of radar images of Delfland catchment area between RT and MFBS datasets for three different timestamps, (a-c) RT radar images of the selected timestamps, (b-d) MFBS radar images of the selected time stamp

For this research project, it is crucial to use real-time data as input for prediction because nowcasting aims at predicting rainfall in the coming 6 hours, while it may take more than 6 hours to produce the MFBS data. So, the low-level processed and real-time RT dataset is used as the primary dataset for our model training and testing. The MFBS dataset can provide a more accurate indication of rainfall, so it is used as the reference for selecting rainfall events in this thesis work. Also, researchers have proposed methods to reduce the bias between the RT dataset and MFBS dataset [15], which makes it possible to compare our model's prediction result with the MFBS dataset. Details are introduced in the following sections.

3.1.2. Data analysis

For pixel-level analysis, the data are selected as follows: first, 60,000 radar images are sampled from 2008-2014. Specifically, starting from 00:00 01/01/2008, one radar image was sampled every hour, repeating 60,000 times to cover all different rainfall situations. The occurrence of different precipitation intensities (with an accuracy of 0.1mm) was counted. The analysis result is shown in figure 3.4.

(a)

Rainfall intensity X	Occurrence	Percent
X <= 0.1mm/h	3723833158	94.7%
0.1mm/h < X <= 1mm/h	165291440	4.2%
1mm/h < X <= 5mm/h	39923265	1.0%
5mm/h < X <= 10mm/h	2355740	0.06%
X > 10mm/h	694446	0.02%

(b)

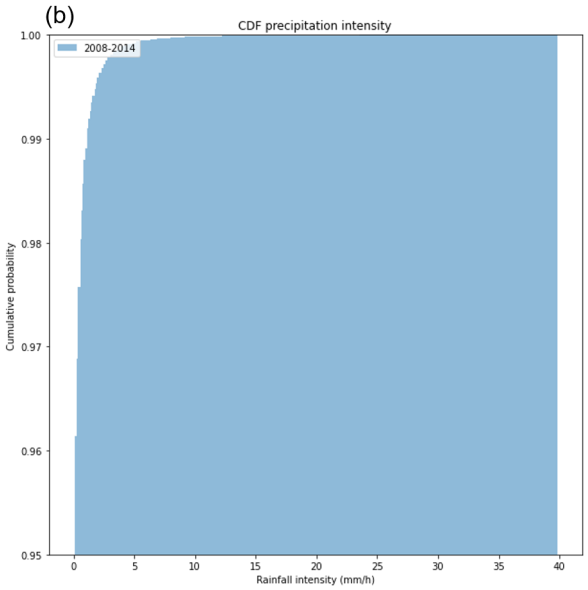

Figure 3.4: Pixel-level analysis result (a) Summary of the occurrence of different types of rainfall in pixels (b) Cumulative probabilistic function of pixel rainfall intensity

Catchment areas are land where runoff collects in specific zones. The nowcasting results of the catchment area are crucial since these results can be used in hydrological modeling for flood early warning. In this thesis, 12 Dutch catchments will be analyzed. The location of these catchment areas is shown on the map in figure 3.5.

Number	Catchment name	Area (km^2)
1	Regge	957
2	Aa	836
3	Delfland	379
4	Reusel	176
5	Linde	150
6	Rijnland	89
7	Roggelsebeek	88
8	Dwarsdiep	83
9	Beemster	71
10	Luntersebeek	63
11	Grote Waterleiding	40
12	Hupsel Brook	6.5

Figure 3.5: The locations of 12 catchments across the Netherlands and their corresponding area

For catchment level analysis, we looped through every possible event starting time so that all 3-hour events between 2008-2014 are examined. The average rainfall accumulation in catchment areas during this three-hour time window is calculated and used as the main indicator of rain intensity for the catchment area. Table 3.6 summarize the analysis result

Average rainfall intensity X	Occurrence	Percent
X <= 0.01mm/h	931117	68.1%
0.01mm/h < X <= 0.1mm/h	137779	10.1%
0.1mm/h < X <= 1mm/h	176938	12.9%
1mm/h < X <= 2mm/h	52492	3.8%
2mm/h < X <= 3mm/h	27415	2.0%
3mm/h < X <= 4mm/h	15358	1.1%
4mm/h < X <= 5mm/h	9126	0.67%
5mm/h < X <= 6mm/h	5447	0.40%
6mm/h < X <= 7mm/h	3745	0.27%
7mm/h < X <= 8mm/h	2630	0.19%
8mm/h < X <= 9mm/h	1511	0.11%
9mm/h < X <= 10mm/h	1078	0.079%
X > 10mm/h	2787	0.20%

Figure 3.6: Catchment-level analysis result

From both pixel level and catchment level analysis, we can reach several conclusions:

1. The distribution of rainfall intensity is highly imbalanced, both distributions have more than 90% of pixels or catchment averaged precipitation smaller than 1mm/h, while the highest value can be larger than 40mm/h for pixel analysis and 20mm/3h for catchment analysis.
2. Because of the highly unbalanced problem, the basic deep learning model may face difficulties fitting the data, and extra techniques are necessary.
3. The distributions still have a relatively (compared with exponential distribution) high probability in its tail part, so a heavy-tailed distribution may be needed to model the rainfall intensity.

3.1.3. CARROTS bias correction [15]

The RT dataset is used as the main dataset in this thesis work. Although the precipitation estimated from the radar reflectivity is inaccurate, it provides a real-time update, which is more important for the nowcasting task. However, because the MFBS dataset provides more accurate information about the rainfall, it is worth comparing our prediction result with the MFBS data. To correct the bias in the RT dataset, researchers proposed the CARROTS correction factor, which can partly correct the bias. The CARROTS factor contains a factor map for every day of a year, and ideally, the bias can be corrected by simply multiplying our prediction with its corresponding CARROTS factor.
The CARROTS factor used in this thesis work is calculated from 2009-2018, and it is calculated as follows:

1. (For both RT and MFBS datasets) For every day in the time range, sum all precipitation maps within a 31 days time window, which includes 15 days before and after the date.
2. Average the 31 days sum over ten year
3. Divide the average of MFBS dataset result by the average of RT dataset pixel by pixel and get the factor map for every day of the year.

Figure 3.7 shows an example of applying the CARROT figure to fix the bias of the RT dataset. The result shows that the CARROTS can partly correct the underestimation problem. The MAE between RT and MFBS radar images drops from 11.45 to 8.36 for this example.

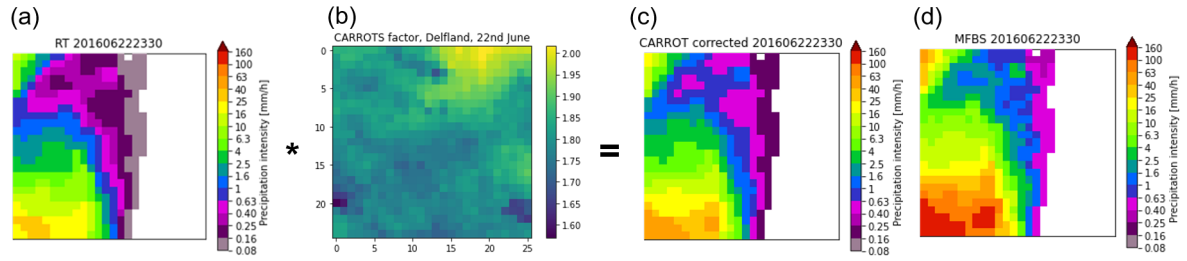

Figure 3.7: An example of correcting bias of RT radar image using CARROTS, (a) The original RT radar image of Delfland catchment with at 23:30 on 22nd June. (b) The CARROTS correction factor for Delfland catchment on 22nd June. (C) The CARROTS corrected the result. (d) The corresponding MFBS radar image

3.2. Problem formulation

The model is expected to fulfill two goals: first, it's expected to output skillful precipitation nowcasting results for the whole Netherlands. Second, it is expected to detect extreme events happening in catchment areas reliably.

For the first goal, this thesis work aims at nowcasting with a maximum lead time of 3 hours and time interval of 30 minutes, so one event contains six frames (T+30, T+60, T+90, T+120, T+150, T+180 minutes), and the radar maps of the previous 90 minutes are used as conditions (T-60, T-30, T minutes). The KNMI dataset provides radar maps with shapes of (765, 700). However, most of the areas on the map are masked, and only the area covered by radars (a circle area with a diameter of around 400km) contains information. Because we focus on rainfall in catchments and the efficiency of training and avoiding including the masked area on radar images, a (256, 256) area (a 256km*256km area on the map) is cropped from the original map and used as our study area. This area covers around 90% of the land area of the Netherlands and also all 12 catchments area completely. The study area is shown in figure 3.8

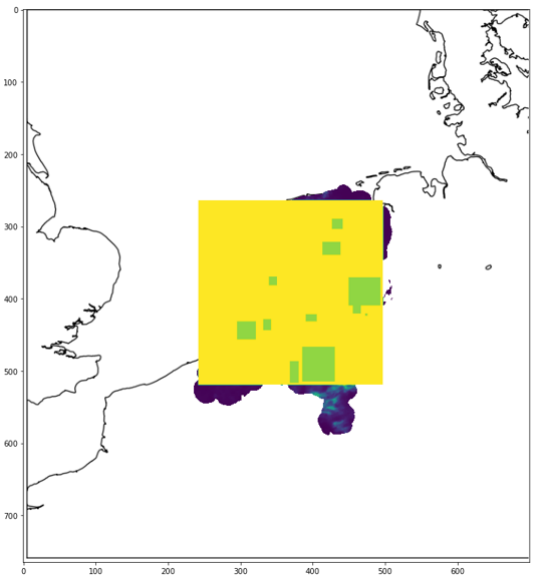

Figure 3.8: The study area for this thesis work (yellow area) and the catchments area (green area) on the Netherlands map

For the second goal, extreme events need to be defined first. The extreme value is defined as values having a significant deviation from the median of its probabilistic distribution. From the analysis in the previous section, it is possible to define the extreme event in two ways: pixel-level extreme event and

catchment-level extreme event.

From the analysis, 10mm/h can be selected as a reasonable extreme threshold for a pixel-level extreme event. However, the detection of such extreme events may not be meaningful for two reasons: first, the data itself is an estimation of the rainfall intensity. It has noise, which makes the detection target itself inaccurate. Second, a single pixel larger than the threshold may not lead to extreme-rainfall-relevant hazards, which are usually caused by heavy and long-term rainfall in certain areas. So, instead, we determined to focus on specific areas' rainfall accumulation over a certain period.

Also, because of the importance of forecasting rainfall intensity for catchment areas, in this study, extreme events are defined as 3-hour events with an average precipitation of a certain catchment area larger than the extreme threshold. The procedures for selecting these thresholds are introduced in the next section. The study areas are the 12 Dutch catchment areas shown in the data analysis section. The pipeline of the system is shown in figure 3.9

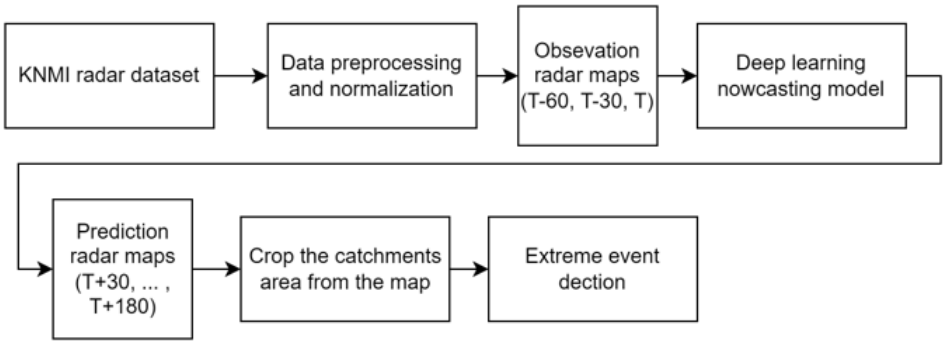

Figure 3.9: The general pipeline of the overall system

3.3. Event selection

In this thesis work, an event that happened in one catchment is defined as an extreme event if the catchment average precipitation accumulation over 3 hours exceeds the catchment's corresponding extreme threshold. The extreme threshold for a certain catchment can be defined as follow:

1. Select one catchment area as the study area
2. Classify all event into no rain event (catchment average precipitation < 0.1mm/3h) or rainy event (>= 0.1mm/3h)
3. Sort the events based on average catchment precipitation, classify the top 1% of the events as an extreme event, the extreme threshold for this catchment is then determined

The analysis example shows that the extreme threshold for Delfland catchment is 9.89mm/3h. With this threshold, it's straightforward to select the extreme events for the Delftland catchment:

1. Loop through every 5 minutes radar data file, use its time as the event starting time, define the next 3 hours as an event
2. Calculate the average precipitation for the catchment during this event
3. Select the event if the average precipitation exceeds the extreme threshold

However, only extreme events are not enough data to train the deep learning model. Similarly, we can define the top 5% of the events as heavy rainy events for the catchment and include them in the training and validation set. Finally, the selected events are split into three parts: events that happened in 2008-2014 are used for training, 2015-2017 are used for validation, and 2018-2020 are used for testing. In total, for the Delfland catchment, 7257 events are selected for training, 3290 events are selected for validation, and 777 events (all extreme) are selected for testing.

4 Methodology

4.1. Proposed Model

The section starts with an overview of the proposed model structure. Then, the two stages of the model are introduced in detail in the second and third subsections.

4.1.1. Model overall architecture

The proposed nowcasting model is based on a recently developed deep generative model for visual synthesis tasks. The model uses a two-stage approach, where the first stage learns to encode the original data into meaningful latent space representation and the second stage learns a probabilistic model for this latent space. This idea was first realized using a Variational Autoencoder (VAE) to learn the data representation in the first stage. Then, a second VAE is used to learn the probabilistic distribution of the latent space representation [8]. This method has shown an advantage over standard approaches (e.g., single VAE) in multiple papers. For our proposed nowcasting model, a Vector Quantization Generative Adversarial Network(VQ-GAN) [10] is used as the first stage to learn the meaningful discrete encoding of the radar map. An autoregressive transformer is used as the second stage to model the distribution of this discrete encoding. The pipeline for using this model is as follows:

- Training process
  - Data prepossessing and normalization
  - Train the VQ-GAN
  - Using the encoder of the trained VQ-GAN, transform all radar maps to corresponding latent space representation
  - Train the transformer using the latent space dataset
- Generating process
  - Data prepossessing and normalization
  - Using encoder of the VQ-GAN, transform condition radar maps to corresponding latent space representation (condition tokens)
  - Create an empty list for prediction tokens
  - (Loop start)
  - Use transformer to get probabilistic distributions for the next tokens (Input: current token list and condition tokens)
  - Sample the prediction token from the probabilistic distribution
  - Append the sample result to the current token list
  - (Loop end)
  - Using decoder of the VQ-GAN, transform prediction tokens to the radar map
  - Map the data back to the original domain

The general structure of this model is shown in figure 4.1

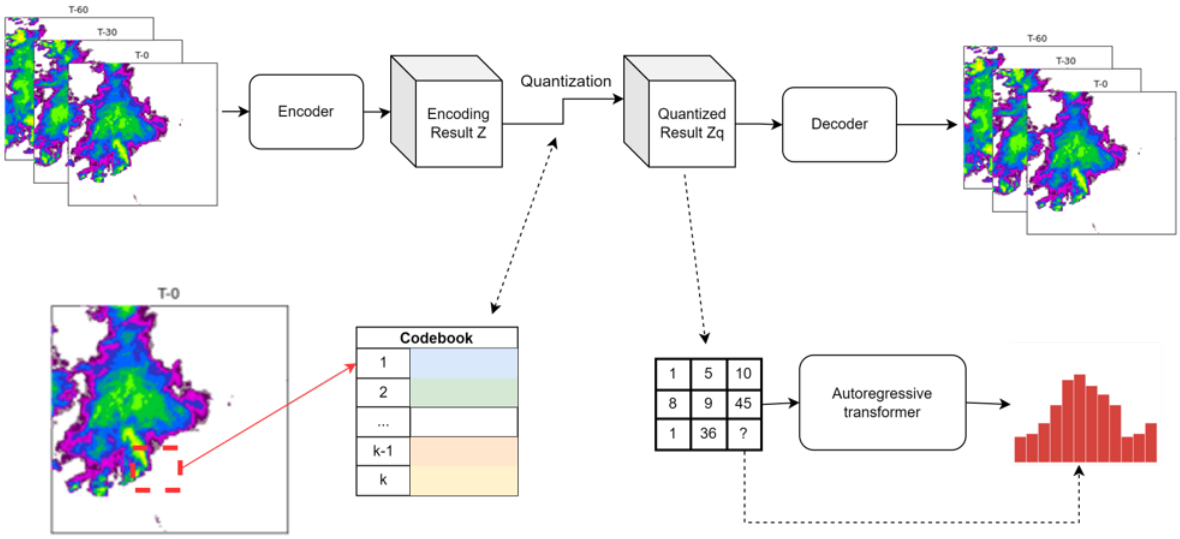

Figure 4.1: The overall structure of the proposed model. In the first stage model, VQ-GAN learns a codebook, and each code (or combination of codes) in the codebook is a representation of a certain pattern on the radar images. The radar data can then be compressed and represented by a sequence of indices, whose composition is modeled subsequently by the autoregressive transformer. When making a prediction, a sequence of condition indices is sent to the transformer. The trained transformer can then generate probabilistic distributions of prediction tokens in an autoregressive way.

4.1.2. First stage: VQ-GAN [10]

VQ-GAN was first proposed in [10] for image generation-related tasks. The goal of using this model in our system is to compress the original radar image into a smaller dimension latent space, which allows for the efficient use of the transformer in the second stage of the model. The model is made up of a Vector Quantization Variational Autoencoder (VQ-VAE) [34] for image reconstruction and a discriminator for adversarial training and sharper reconstruction result. The reconstruction result is shown in 4.2.

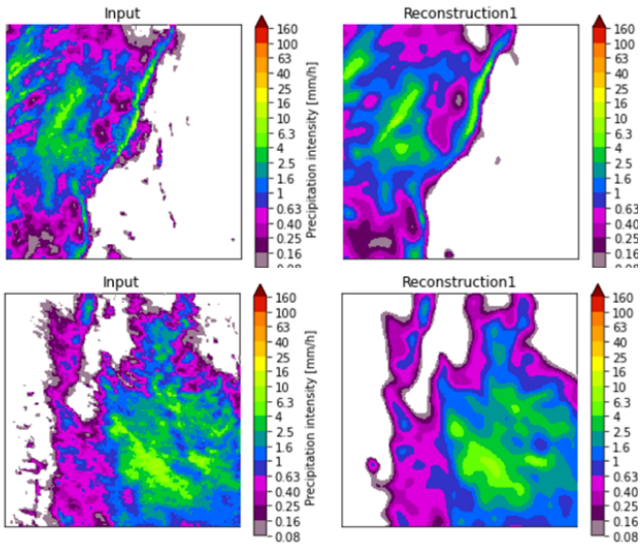

Figure 4.2: Examples of reconstruction of precipitation fields produced by VQ-GAN (Left column: original precipitation fields; right column: corresponding reconstructed precipitation fields)

VQ-VAE (Vector Quantization Variational Autoencoder)

The VQ-VAE model contains three main components: an encoder, a decoder, and a codebook. The high-level structure of the whole model is shown in figure 4.3, and the structure of the decoder and encoder are shown in 4.4. The encoder first encodes the input data into a smaller dimension latent space. Second, the encoding result is mapped to one of the codebook's codes (or tokens) using vector quantization. Third, the decoder can decode the tokens to the original precipitation field. The detail of the forward path's data dimensions are as follow:
(b: batch size; h: height; w: width; t: time; k: code-book size (total number of possible tokens); d: token dimension)

1. Input size $I : (b, h, w, t)$
2. Encoded size $Encode(i) : (b, h', w', t, d)$(each time frame are encoded separately)
3. Code-book size $Dict : (k, d)$
4. Rearrange $Encode(i)$ as tokens: $(b, (h' * w' * t), d)$
5. Calculate the token distance to each k class in code-book: $(b, (h' * w' * t), k)$
6. Find the smallest distance index: $(b, (h' * w' * t), 1)$
7. Assign corresponding code from the code-book and rearrange: $Dict(Encode(i)) = (b, h', w', t, d)$
8. Decode: $Decode(Dict(Encode(i))) = (b, h, w, t)$

The loss function for VQ-VAE is as shown below:

$$L_{VQ} = \|I - \hat{I}\|_2^2 + \|sg[E(I)] - B[z]\|_2^2 + \|E(I) - sg[B[z]]\|_2^2 \tag{4.1}$$

The first term is the reconstruction loss between the input I and the reconstructed input $\hat{I}$. When backpropagate, it will skip the codebook part (argmin operation in the vector quantization process is not differential, so the gradient before and after codebook operations are assumed to be unchanged).

The second term is used to update the code book and is called the "commitment loss". The sg term means stop gradient operation. This loss term penalizes the difference between the encoder output and codebook. The encoder is stop gradient, so only the codebook will be updated for this term.

The third term is the same as the second but with a stop gradient operation on the codebook, so only the encoder will be updated to get encoder output as close to embedding in the codebook as possible.

Discriminator

Researchers have shown evidence that the use of discriminator and adversarial training allows the model to generate sharper image [10]. Multiple papers have proposed to combine VAE with an additional discriminator to achieve better generation or reconstruction performance. Similar to [10], a CNN-based discriminator is introduced for our VQ-VAE, and the new model is called VQ-GAN. The model's high-level structure of the discriminator is shown in 4.5. Besides, perceptual loss [17] is used as a second reconstruction loss to keep better perceptual quality for reconstruction. In this way, the loss function changed, with the discriminator loss maximizing $L_D = \log D(I) + \log(1 - D(\hat{I}))$ and generator loss minimizing $L_G = L_{VQ} + \log(1 - D(\hat{I}))$.
The complete objective function for this first stage model is than:

$$\begin{aligned} \arg\min_{E,G,\mathcal{Z}} \max_{D} \mathbb{E}_{x\sim p(x)} [&\mathcal{L}_{\text{VQ}}(E, G, \mathcal{Z}) \\ &+ \lambda \mathcal{L}_{\text{GAN}}(\{E, G, \mathcal{Z}\}, D)] \end{aligned} \tag{4.2}$$

where E, G, Z, and D are encoder, decoder, the codebook, and the discriminator, respectively and λ is an adaptive weight, which is calculated as:

$$\lambda = \frac{\nabla_{G_L} [\mathcal{L}_{\text{rec}}]}{\nabla_{G_L} [\mathcal{L}_{\text{GAN}}] + \delta} \tag{4.3}$$

Where $\nabla_{G_L}[L]$ is the gradient of the loss function concerning the last layer of the generator, and δ is a small number for stability.

Model Structure

The structure of the overall model is presented in figure 4.3. Figure 4.4 presents the decoder and encoder structure. The structure of the discriminator is presented in figure 4.5.

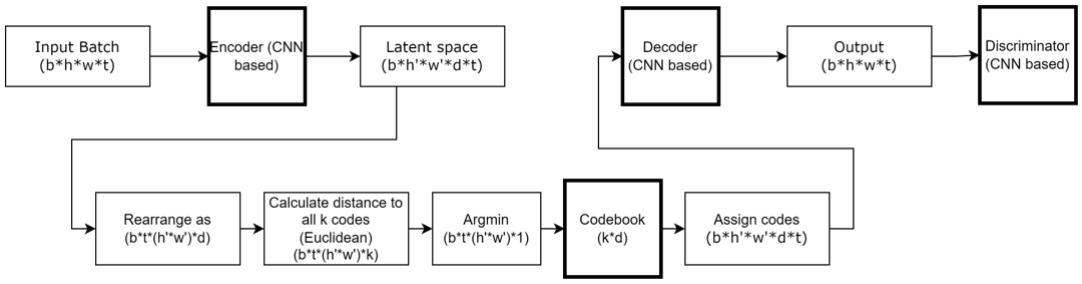

Figure 4.3: The overall structure of VQGAN

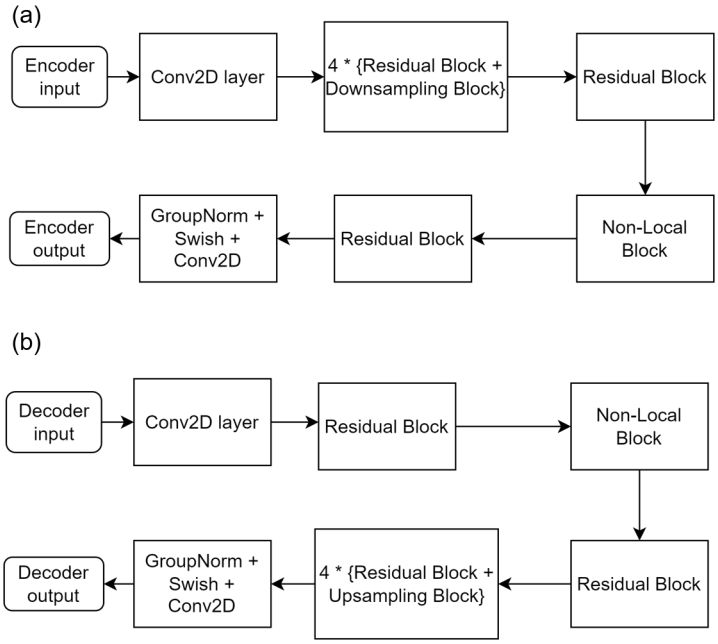

Figure 4.4: High-level structure of the encode and decoder of VQGAN

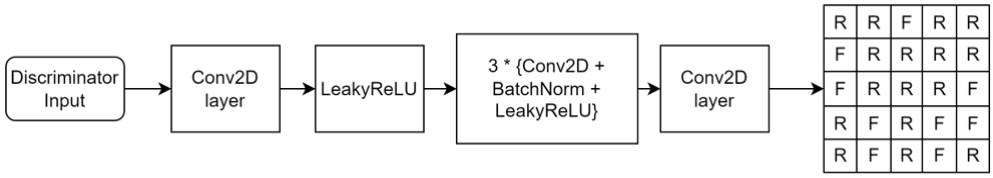

Figure 4.5: High-level structure of the discriminator of VQGAN. The discriminator is a patch-based discriminator, which outputs a 5*5 metrics of fake/real probability

4.1.3. Second stage: autoregressive transformer

After the stage one model is trained and fixed, the radar map data can now be encoded into a smaller dimension latent space and represented by a sequence of tokens (from the codebook), which is used as the input of the autoregressive transformer in the second stage.

Autoregressive transformer

A transformer is a type of deep learning network which solely uses the attention mechanism for interaction between inputs [35]. The use of attention allows the network to capture longer-term relationships between inputs. Transformer is originally applied in sequential task like language and audio [25][35], but in recent years, multiple research papers have proven its promising potential in image and video tasks [21][20]. Typically, a transformer contains an encoder and decoder, both of which are made up of multiple layers of attention mechanism. The attention contains three main components: query(Q), key(K), and value(V). The similarity is calculated between Q and K. Then, based on the similarity, different weights are assigned for different positions in V. The weighted average of V is the final output. This operation is expressed as the equation below:

$$Attn(Q,K,V) = softmax(\frac{QK^t}{\sqrt{d_k}})V \tag{4.4}$$

Where d_k is the length of the query. In most cases, self-attention is used [35], which means that all Q, K, and V are calculated from the same sequence but with different weights. From the equation, the attention is computed from the product between all possible pairs in a sequence, so the computational time increases quadratically with sequence length. This finding emphasizes the importance of the first stage, which largely compresses the high-dimension radar data into a sequence of tokens.

The autoregressive transformer is a variant of the original transformer and has been widely used in generation tasks (e.g., [25][38]). In general, for an autoregressive transformer, a sequence of tokens is fed into multiple transformer blocks, which can output a probability distribution of the next token. The main difference compared with the original transformer is that: first, in terms of the model architecture, it only contains the decoder part of the original transformer. Second, in terms of the attention mechanism, causal attention is used instead, which masks all tokens behind the current token.

3D nearby self/cross attention layer (3DNA) [37]

Instead of the full attention used for the original transformer, a sparse attention layer called 3DNA is used as the attention layer for our nowcasting model. The 3DNA allows more efficient training and takes advantage of the 3D shape of video data, making it a suitable choice for our task. The idea is for 3D tensor of shape (h, w, t), each token of location (i, j, k) only do attention to tokens within the cube of size (e^h, e^w, e^t) around this location. In the case of self-attention, the operation can be expressed as:

$$N^{(i,j,k)} = \left\{ X_{abc} || a - i \,| \leq e^h, | \, b - j \, | \leq e^w, | \, c - k \, | \leq e^s \right\} \tag{4.5}$$

$$\begin{aligned} Q^{(i,j,k)} &= X^{(i,j,k)} W^Q \\ K^{(i,j,k)} &= N^{(i,j,k)} W^K \\ V^{(i,j,k)} &= N^{(i,j,k)} W^V \\ y_{ijk} &= softmax\left(\frac{\left(Q^{(i,j,k)}\right) K^{(i,j,k)T}}{\sqrt{d^{in}}} \right) V^{(i,j,k)} \end{aligned} \tag{4.6}$$

In the equations, $N^{(i,j,k)}$ is a set of neighbourhood tokens around the tokens with location (i, j, k) and (e^h, e^w, e^s) is the dimension of the neighbourhood cube. Because casual attention is used, only tokens within this cube and tokens before the current token in the sequence are included in N. Ws are weights for query, key, and value, with a dimension of (d, d'). In this way, query $Q : (1, d')$, key and value $K/V : (n, d')$ (n: no. of neighbors). It's a multi-head attention operation with h heads, so the concatenated heads results are $(1, h * d')$, by multiplying the multi-head weight $W^O : (h * d', d'')$, we can get the final y_{ijk} with dimension $(1, d'')$ In case of cross attention, Q remains the same, K and V's $N^{(i,j,k)}$ get from condition tensor instead.
An example of 3DNA is shown in figure 4.6.

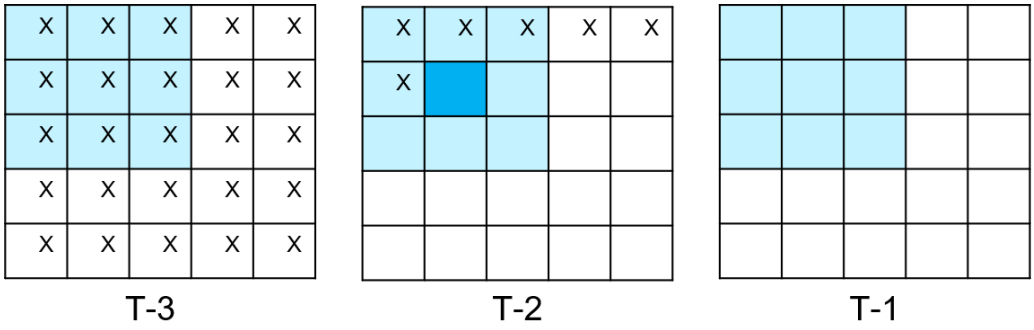

Figure 4.6: An example of 3DNA mechanism. The current token is marked with dark blue. For 3DNA with a kernel size of 3, the attention result of the current token is calculated from values in nearby tokens (marked with light blue). Because of causal attention, only the tokens before the current tokens in the sequence are used (marked with "X"). Overall, tokens marked with both "X" and light blue are used for attention calculation.

Transformer details

The prediction output is first encoded into its latent space representation and used as the model's input. The prediction tokens are right-shifted, and all tokens behind the current token are masked. So, when computing casual attention, the current token can only "see" the previous tokens in the sequence. Then it is fed into L layers of the 3DNA block where the sequence computes its cross attention to the condition tokens and self-attention to the output sequence of the previous layer:

$$\begin{aligned} Y_{ijk}^{(l)} = 3DNA\left(Y_{<i,<j,<k}^{(l-1)}, Y_{<i,<j,<k}^{(l-1)}\right) \\ + 3DNA\left(Y_{<i,<j,<k}^{(l-1)}, C\right) \end{aligned} \tag{4.7}$$

In the equations, Y^l is the output of the l layer, and C is the condition's latent space (observed radar precipitation map).
The final output is a classification result, with each final output token being classified into one of the vectors in the first stage codebook. The cross-entropy loss can be used as the loss function with the prediction result as input and ground truth observation's latent space representation as the target. The details of the forward path is shown below:
(b: batch size; h: height; w: width; t: time; k: codebook size; d: token dimension; t': The prediction time; d': inner dimension)

1. The observation latent space as condition: $C = (b, h', w', t, 1)$
2. The predication latent space $Y = (b, h', w', t', 1)$
3. Flatten the input into a sequence and shift one block to the right: $Y = (b, (h' * w' * t'), 1)$
4. Embed both $(b, (h' * w' * t), d)$ and $Y : (b, (h' * w' * t'), d)$
5. 3D nearby cross and self attention layers output for prediction $Y^{(L)} = (b, (h' * w' * t'), d')$, where d' is the inner dimension of attention block
6. Input $Y^{(L)}$ to a fully connected layer: $FC(Y^{(L)}) = (b, (h' * w' * t'), k)$ with $FC = (d', k)$ and k the dimension of the code-book
7. Calculate the cross-entropy loss, with transformer output as input and transformer input (prediction token Y) as the target.

In the case of generation, the process is a bit different from the training stage. Instead of using the prediction's latent space representation as input, an empty token list is used as the model's first input. The model will then output a probabilistic distribution for the first token, sample the token from the distribution, and append it to the token list. Finally, the new token list is used as the model's input, and the process repeats until all needed tokens are acquired. The main difference is shown in figures 4.7 and 4.8:

Model structure

The structures of the overall model is shown in figure 4.7 (Training stage) and figure 4.8 (Generation stage).

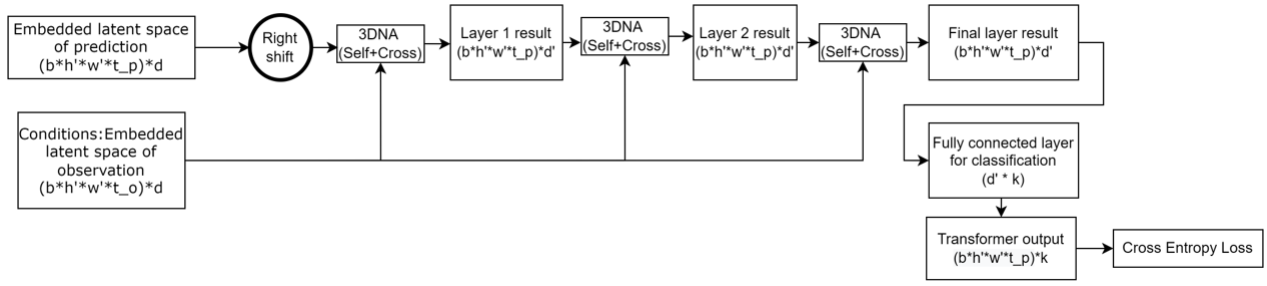

Figure 4.7: The overall structure of the transformer (Training stage)

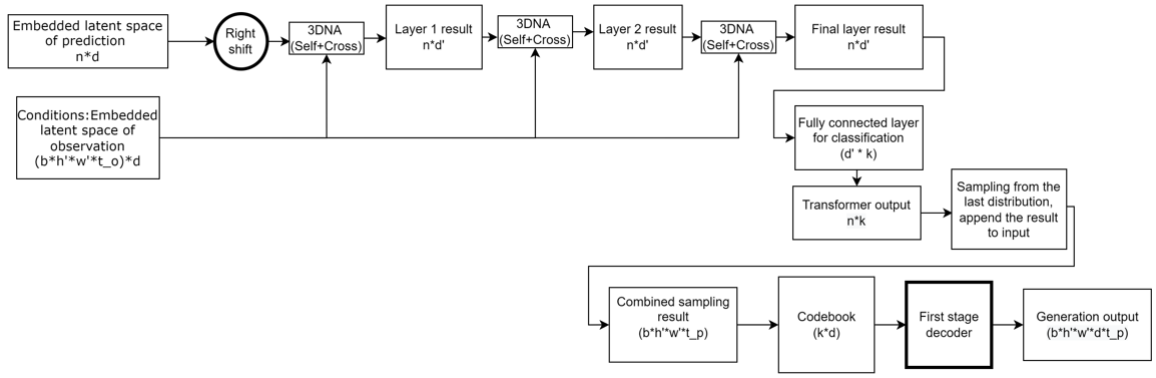

Figure 4.8: The overall structure of the transformer (Generation stage)

4.2. Handling extreme events

Problem

For the latent space representation of our radar dataset, tokens (or tokens combination) represent no rain, light, heavy, and extreme rain area on the radar precipitation map. Nevertheless, because of the nature of rainfall, the occurrence of no rain and light rain tokens is much larger than heavy rain. Even for the extreme event selected, it is common to have more than 50% of the area not rainy. This finding means that we are training the transformer with a highly unbalanced dataset. From a probabilistic point of view, minimizing the cross entropy loss is equivalent to the optimization problem of maximizing the likelihood as follows:

$$\max_{\theta} \prod_{n=1}^{h*w*t} P\left(y_n \mid y_{<n}, c, \theta\right) \tag{4.8}$$

where θ is the model's parameter, $h * w * t$ is the total number of tokens for one event.

Because there are insufficient or no samples for extreme precipitation patterns from the training set, we are expected to see a decrease in performance for these extreme-related tokens. The model's output distribution will also be biased towards the tokens with dominant occurrence (tokens representing no or light rain).

Solution 1: Extreme value loss (EVL) [9]

To better model the extreme events for the nowcasting task, EVL is proposed to incorporate with the transformer loss. Extreme value loss is first proposed for modeling extreme events in time series. The loss function is based on the extreme value theory theorem to model the distribution's tail part. Specifically, from the extreme value theory, the tail distribution of real-world data y can be modeled as follow [11][36]:

$$1 - F(y) \approx (1 - F(\xi)) \left[1 - \log G\left(\frac{y - \xi}{f(\xi)}\right)\right], y > \xi \tag{4.9}$$

where ξ is the threshold between normal and extreme value, function $G(x)$ is the generalized extreme value distribution (GEVD), function $F(x)$ is the probability of existing a value larger than x. Assuming the detection of extreme events as a binary classification task, the cross-entropy loss can then be used to penalize detecting extreme events as non-extreme and false alarm cases. In addition, the term $\frac{y-\xi}{f(\xi)}$ can be approximated by an extreme indicator u, which indicates the probability of the current predicted event being an extreme event. The approximated extreme probability can then be used as the weights for the cross entropy loss. The loss can be approximated and expressed as follow:

$$\begin{aligned} EVL\left(u_t\right) = & -\left(1 - P\left(v_t = 1\right)\right)\left[\log G\left(u_t\right)\right] v_t \log\left(u_t\right) \\ & -\left(1 - P\left(v_t = 0\right)\right)\left[\log G\left(1 - u_t\right)\right]\left(1 - v_t\right) \log\left(1 - u_t\right) \\ = & -\beta_0 \left[1 - \frac{u_t}{\gamma}\right]^{\gamma} v_t \log\left(u_t\right) \\ & -\beta_1 \left[1 - \frac{1 - u_t}{\gamma}\right]^{\gamma} \left(1 - v_t\right) \log\left(1 - u_t\right) \end{aligned} \tag{4.10}$$

where β_0 and β_1 are the proportions of the normal and extreme events in the training set, respectively. v_t is the ground truth indicator which will be 1 for extreme and 0 for normal. γ is a hyper-parameter.

To implement this loss function, extreme needs to be defined for our dataset first. Because the transformer works in the latent space of the first stage model and the extreme events are defined by the area averaged precipitation accumulation, we can classify the tokens into extreme or non-extreme tokens (each token has its corresponding area with different levels of rainfall intensity on the radar precipitation field). Extreme tokens meet the following requirement: first, they have low occurrence when considering the whole study area. Second, have a high occurrence when considering an extreme event in the catchment area. Third, it can be decoded to high-intensity rainfall. An example of the extreme token is shown in figure 4.9. This example extreme token can produce a high precipitation pattern in the radar image from the figure.

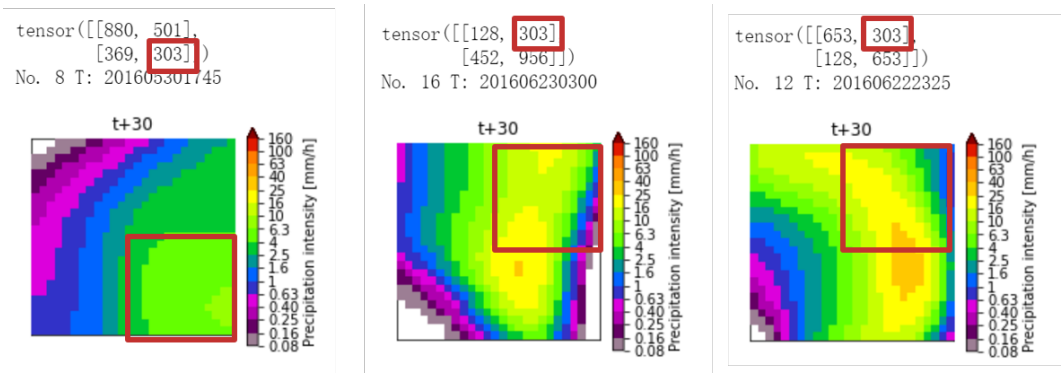

Figure 4.9: An example of selected extreme tokens: 303. The red box roughly marks the area represented by token 303 in the case of different combinations.

In this way, 15 extreme tokens are selected. The output of the autoregressive transformer is a probability distribution, so the extreme indicator can be computed by summing up the corresponding normalized probability of these extreme tokens.

Solution 2: class weight

A standard solution to the dataset unbalance problem is using different class weights, emphasizing the less occurred extreme tokens in the loss function by assigning a larger weight. In this case, a weighted cross entropy loss can be applied, with weight inversely proportional to the occurrence of each token in the training set. This method is also applied and compared with the EVL-trained model.

Solution 3: post processing

In [7], a post-processing method for nowcasting is proposed to emphasize the high precipitation pixels. The method is expressed as the equation below:

$$TP[i][j] = \left(1 + a\left(\frac{RP[i][j]}{\max(RP)}\right)^{b}\right) * RP[i][j] \tag{4.11}$$

where TP and RP are processed and unprocessed predictions, respectively. Moreover, (i,j) indicates the location of the pixel. a and b are two parameters whose values are determined to reach the maximum Gilbert Skill Score (GSS) on the validation set. a is set to 0.66, and b is set to 0.81 for this thesis.

This post-processing method can effectively increase the detection rate of high precipitation pixels while increasing the false alarm cases. So, the post-processing is only applied for the averaged result where the false alarm rate is significantly improved.

4.3. Experiment configuration

This section includes the main configuration of the models for the experiment, more details about the experiment are included in the next chapter.

1. VQ-GAN model configuration:
    - Dimension of codebook: 1024
    - Token dimension: 256
    - Number of down-sampling layer: 4 (Compression rate: 16)
2. VQ-GAN training configuration:
    - Learning rate: 1e-6
    - VQ decay rate (Update speed of codebook): 0.5
    - Batch size: 64
    - Weight decay: 0.1
3. Autoregressive transformer model configuration:
    - Embedding dimension: 512
    - Number of attention layer: 12
    - Number of attention head: 8
    - 3DNA kernel size (self-attention): (7, 3, 3)
    - 3DNA kernel size (cross-attention): (5, 3, 3)
4. Autoregressive transformer training configuration:
    - Learning rate: 1e-4
    - Batch size: 64
    - Weight decay: 0.1
    - Drop out rate (for both attention and fully connected layer): 0.2

The training and experiments are conducted on an NVIDIA RTX A6000 GPU.

4.4. Verification

This section presents the verification method used for the experiments. The first two subsections introduce different metrics for evaluating the nowcasting performance for the whole Netherlands. The third subsection introduces the plan to evaluate the performance of extreme event detection.

4.4.1. Continuous metrics

Pearson's correlation (PCC)

PCC measures the linear correlation between prediction and observation. It is calculated as follows. A larger PCC means a better result.

$$PCC = \frac{1}{N_f} \sum_{i=1}^{N_f} \frac{(F_i - \mu_F)(O_i - \mu_O)}{\sigma_F \sigma_O} \tag{4.12}$$

where F_i and O_i are the rainfall amount at a certain cell of the prediction and observation map, respectively. μ_F and μ_O are the mean rainfall amount of prediction and observation radar map. σ_F and σ_O are the standard deviation of rainfall amount of prediction and observation radar map. N_f is the total number of pixels in the predicted radar map at a certain lead time. The PCC is calculated for every lead time in the prediction result. Generally, predictions with PCC larger than a threshold (usually set to around 0.37) are considered skillful. In this way, the researcher uses this metric to determine the maximum skillful lead time of the model.

Mean absolute error (MAE)

MAE is a commonly used and straightforward metric for nowcasting task. It can be calculated as follow. A smaller MAE means better performance.

$$MAE = \frac{\sum_{i=1}^{N_f} |F_i - O_i|}{N_f} \tag{4.13}$$

4.4.2. Categorical metrics

To calculate categorical metrics, every pixel on the prediction map and observation map is first classified into either positive (larger or equal) or negative (smaller) based on a given threshold. Then, the pixels are classified into one of the four categories below:

1. H: true positive, observation and prediction are both positive
2. M: false negative, observation is positive, but prediction is negative
3. F: false positive, observation is negative but the prediction is positive
4. R: true negative, observation, and prediction are both negative

Critical success index (CSI)

CSI is a popularly used metric in the nowcasting community. It aims to give a summary of the binary classification performance since it rewards precision and penalize false alarm at the same time. The metric is calculated as follows. Higher CSI means better performance.

$$CSI = \frac{H}{H + F + M} \tag{4.14}$$

False alarm ratio (FAR)

FAR is also an important metric for binary classification performance and is often used in weather forecasting. It indicates the accuracy of alarms in predictions. The metric is calculated as follows. Lower FAR means better performance.

$$FAR = \frac{F}{F + H} \tag{4.15}$$

4.4.3. Spatial metric

Fractions skill score (FSS)

FSS is a spatial verification metric used to assess the performance of precipitation forecasts by measuring the error in the placement of the rain. Different length scale n can be chosen for this metric. A larger n means a more extensive area used for the verification, which usually leads to better results [27]. FSS range from 0 to 1, and a larger FSS means better performance. The FSS is calculated as follows:

$$FSS = 1 - \frac{MSE(n)}{MSE_{ref}(n)} \tag{4.16}$$

where MSE(n) is the mean square error between observation and prediction for length scale n. The reference MSE is the largest MSE can observation and prediction obtained for length scale n. It is calculated as follows:

$$MSE_{\text{ref}}(n) = \frac{1}{N_x N_y}\left[\sum_{i=1}^{N_x}\sum_{j=1}^{N_y} O_{i,j}^2(n) + \sum_{i=1}^{N_x}\sum_{j=1}^{N_y} F_{i,j}^2(n)\right] \tag{4.17}$$

Where N_x and N_y are the number of columns and rows in the observation and prediction radar map. $F_{i,j}^2(n)$ and $O_{i,j}^2(n)$ are the prediction and observation's fractions of surrounding points (up to n) larger than the rainfall threshold for grid cell (i, j). They can be expressed as follows:

$$O_{i,j}^2(n) = \frac{1}{n^2}\sum_{k=1}^{n}\sum_{l=1}^{n} I_O\left[i + k - 1 - \frac{n-1}{2}, j + l - 1 - \frac{n-1}{2}\right] \tag{4.18}$$

where I_O is a binary field if rainfall on the map exceeds the given threshold. The sum of this field is the number of pixels exceeding the rainfall threshold in this grid cell. Divide the sum by the area of the grid cell (n^2), and the fraction $O_{i,j}^2(n)$ can be calculated.
Usually, predictions can be considered skillful predictions when FSS is larger than $0.5 + \frac{f_0}{2}$, where f_0 is the domain averaged rainfall fraction of observations.

4.4.4. Catchment verification

To evaluate the model performance in detecting extreme events that happened in the catchment, verification methods for catchment are introduced in this section. Extreme event is also a binary classification problem. Similar to the categorical metrics, events can be classified into H, M, F, and R based on the extreme thresholds for catchment and the metrics CSI and FAR used for verification. Besides, an additional verification method is used by setting different thresholds for extreme events: the ROC curve.

Receiver Operating Characteristic (ROC) Curve

To draw the ROC curve, two metrics must be calculated first: hit rate (HR) and false alarm rate (FA). They are calculated as follows:

$$HR = \frac{H}{H + M} \tag{4.19}$$

$$FA = \frac{F}{F + R} \tag{4.20}$$

Then, different sets of HR and FA can be achieved by setting different thresholds. The curve can be drawn by putting and connecting these sets on a 2D coordinate with FA as X-axis and HR as Y-axis. ROC is a convenient tool to evaluate and compare different models' detection/classification performance. Usually, the area under the ROC curve (AUC) is used as the indicator for detection ability. A larger AUC represents better performance. An example of the ROC curve is shown below:

5

Experiments and results

Based on the objectives of the project and pipeline of the system shown in figure 3.9, the evaluation results can be split into two steps: nowcasting performance evaluation and extreme event detection ability evaluation.

The model output and the nowcasting results for the whole study area are evaluated in the first part. The evaluation is based on various commonly used metrics (MAE, PCC, CSI, FAR, and FSS) for the nowcasting task to analyze the result from different angles.

To detect extreme event, in the second part, the 3-hour catchment average precipitation accumulation is estimated from the previous nowcasting result. This estimation is compared with the corresponding extreme threshold. Then, each event is classified into one of the four cases (true/false positive/negative) based on the comparison. Finally, various metrics commonly used for binary classification (HR, FA, CSI, FAR, AUC of ROC curve) are used to evaluate the extreme events detection ability. For both parts, PySTEPs performance is used as a benchmark, the detailed configuration of PySTEPs is introduced in chapter 2, and the model is fed with the same input as the deep learning models.

5.1. Nowcasting performance

Two experiments are conducted in this section: the effect of the loss function and the effect of averaging and post-processing. Different loss functions are as follows:

1. Originally, cross-entropy loss (CE) is used for token classification when training the autoregressive transformer. However, the imbalance shown in the original data analysis also leads to imbalance in the latent space tokens and thus poor classification performance.
2. A classic solution to dataset imbalance is class weight, which assigns a higher weight to the minority class and a lower weight to the majority class. In our case, a weighted cross entropy loss (WCE) is applied to replace CE with weights inversely proportionate with their corresponding occurrence.
3. WCE is usually applied to solve small imbalance problems (e.g., 1:5 between minority class and majority class) but our dataset is highly imbalance (e.g., can be 1:10000). To improve this, we further explore the possibility of incorporating extreme value loss (EVL) with our model.

The first experiment in this section studies the effect of applying the different loss functions mentioned above. Besides the loss functions, we also explore the possibility of transferring techniques used in PySTEPS or other deep learning nowcasting models to our model. These techniques are as follows:

1. The result of PySTEPS's ensemble prediction is generated by blending different ensemble members to minimize the uncertainty in rainfall prediction. Our autoregressive transformer can output probabilistic distributions. The results are then generated by sampling from these distributions. So, similar to PySTEPS, the result of generation is a bit different every time, and we can average these results for better performance.

2. Post-processing techniques are considered. In PySTEPs, the post-processing method matches the data distribution of the prediction to the data distribution of the last observation precipitation field. We find this method not work well for our model, so another post-processing method introduced in chapter 4 is applied. The method usually increases the overall rainfall intensity and thus increase both pixel-level CSI and FAR score. The increase of FAR is not acceptable for some non-averaging results considering their initially high FAR. However, as shown later in the section, averaging can decrease the FAR and the overall rainfall intensity, making it feasible to apply post-processing techniques.

The effect of these two techniques is studied in the second subsection.

5.1.1. Effect of different loss functions

In this section, the model trained with different loss functions is compared. These loss functions include CE, WCE, and two types of EVL configurations.
The WCE and CE loss functions are:

$$L_{CE} = -\sum_i g_i ln(p_i) \tag{5.1}$$

$$L_{WCE} = -\sum_i w_i g_i ln(p_i) \tag{5.2}$$

where g, p, w are ground truth (one-hot encoded), softmax probability, and weights for token i, respectively. The first type of EVL is marked as "EVL" in the result, and the loss function is:

$$\begin{aligned} L_{EVL} =& L_{CE} + \lambda * (-\beta_0 \left[1 - \frac{u_t}{\gamma}\right]^{\gamma} v_t \log(u_t) \\ & - \beta_1 \left[1 - \frac{1-u_t}{\gamma}\right]^{\gamma} (1-v_t) \log(1-u_t)) \end{aligned} \tag{5.3}$$

For experiment, parameter γ is set to 1; β_0 and β_1 are set to 0.95 and 0.05 respectively. The weight parameter λ is set to different values, and its effects are studied.

The second type is marked as "EVL FA" in the result. In addition to the original "EVL", this type adds an additional term penalizing the wrong classification within the selected extreme tokens. When the current token belongs to extreme tokens, the loss function is shown below:

$$L_{EVL\_FA} = L_{EVL} - \sum_i g_i^e ln(p_i^e) \tag{5.4}$$

where g^e is the one-hot encoding of extreme tokens and p^e is the softmax probability of these extreme tokens.

The evaluation results is shown in figure 5.1, 5.2 and 5.3. The results show the relationship of the metric scores with lead time. Usually, the nowcasting performance will decrease with increasing lead time.

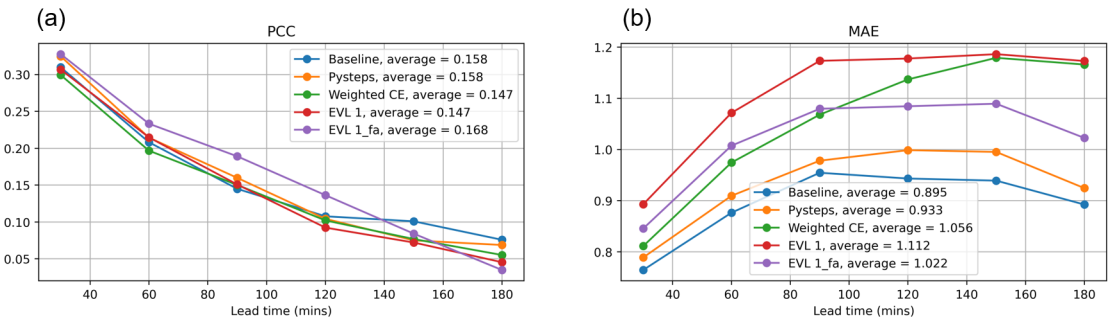

Figure 5.1: 3-hour nowcasting performance verification: continuous metrics (sub-figure a for PCC and sub-figure b for MAE). Relationship between leading time and metric scores, with 3-hour, averaged scores shown in the legend (Pixel-level evaluation)

Figure 5.1 shows the results of continuous metrics PCC and MAE. Usually, when PCC is larger than $\frac{1}{e}$, the prediction is considered skillful. This way, the maximum skillful lead time (or decorrelation time) can be estimated. From the PCC result, most models have a maximum skillful lead time shorter than 30 minutes because of the relatively small studying area and the short event duration. ”EVL FA” and ”PySTEPS” shows better performance, with PySTEPS reaching a decorrelation time of roughly 35 minutes and EVL_FA of around 40 minutes.

For MAE, typically, more significant rainfall intensity leads to larger MAE. High MAE is usually caused by overestimation rather than underestimation in precipitation nowcasting. So, from the MAE result, compared with the PySTEPS, the CE model has a lower MAE, which correctly indicates its prediction result: the precipitation field in the CE model tends to show dissipation of the precipitation field faster than other models. While on the other hand, the EVL and WCE models, compared with PySTEPS, tend to estimate larger rainfall intensity and thus also have more overestimated pixels and larger MAE.

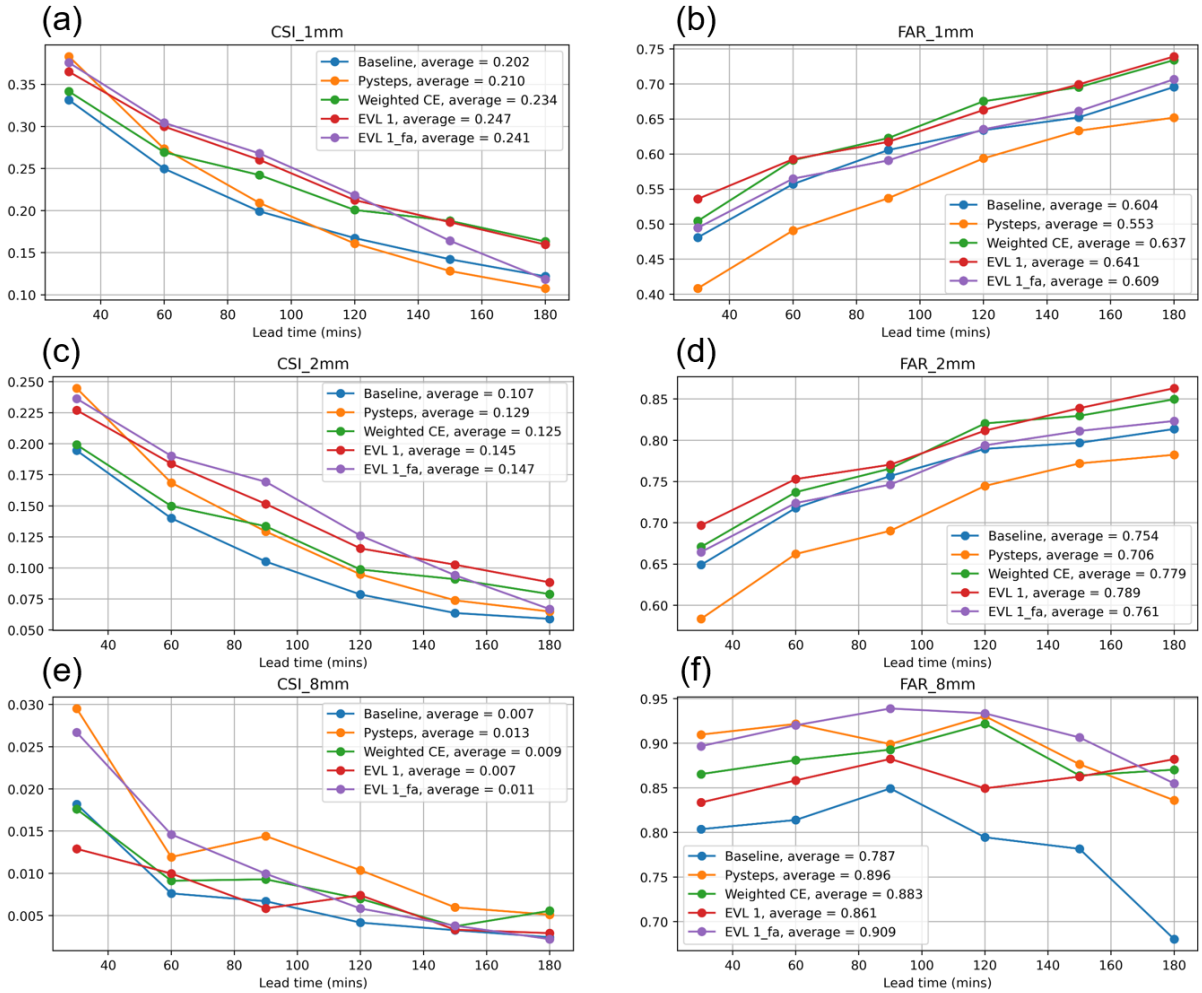

Figure 5.2: 3-hour nowcasting performance verification: categorical scores (CSI and FAR with different thresholds: a,b for 1mm; c, d for 2mm and e, f for 8mm). Relation between lead time and metric scores, with 3-hour averaged scores shown in the legend (Pixel-level evaluation)

Figure 5.2 shows the result of two categorical metrics under different thresholds. In the meteorological community, CSI is popularly used as a summary for the binary classification (exceeding the threshold or not) ability in nowcasting results. Besides CSI, FAR is also used to view the detection ability from another angle. The main aim is to keep the FAR in a reasonable range while at the same time increasing CSI. Different thresholds are used to evaluate the detection performance of a different kinds of rainfall intensity.

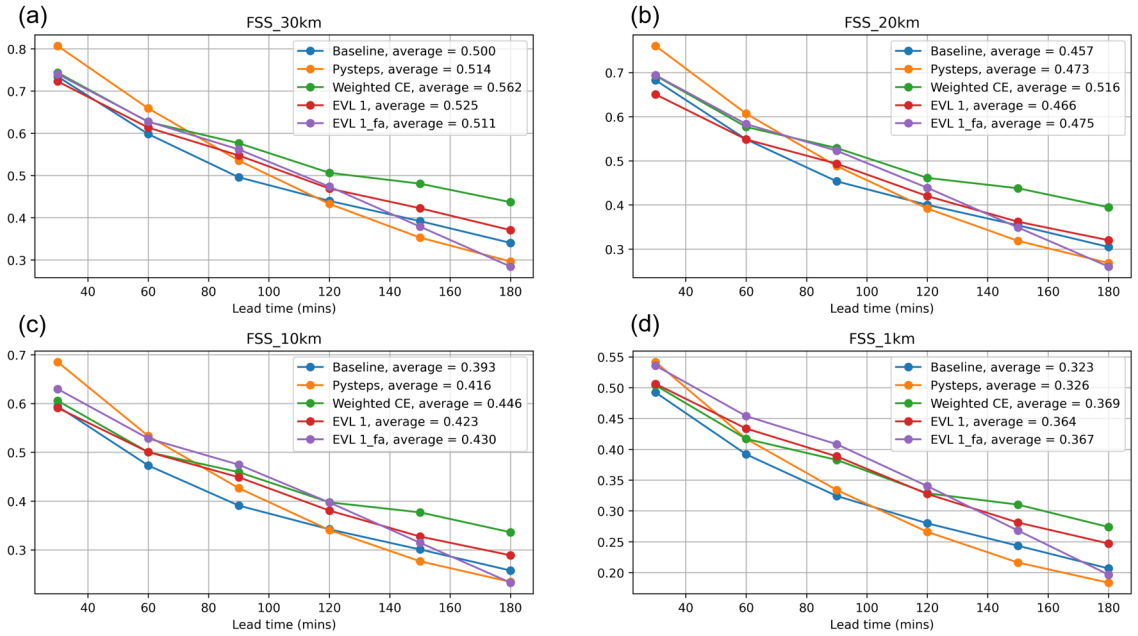

Figure 5.3: 3-hour nowcasting performance verification: spatial scores (FSS with different length scales: sub-figures a, b, c, d representing 30km, 20km, 10km, 1km respectively). Relation between lead time and metric scores, with 3-hour averaged scores shown in the legend (Pixel-level evaluation)

Figure 5.3 shows the spatial verification result, where FSS of different length scales are used. A larger length scale usually has a larger FSS. Intuitively, it means that when we upscale the prediction result to a coarser resolution, there is less error in predicting precipitation field location. A FSS larger than $0.5 + \frac{1}{f_0}$ is considered as skillful, where f_0 is the random forecast skill. For example, for 60 minute lead time, we can conclude from figure 5.3 (c) and (b) that the minimal skillful length scale for all models is between 10 and 20km. This result is also useful for catchment analysis. For example, when considering a catchment of 10km by 10km means that the maximum skillful lead time will be less than 60 minutes if we upscale the prediction from 1km to this catchment level. While for larger catchments (e.g., the largest catchment Regge, with roughly 30 by 30km area), a maximum skillful lead time larger than 60 minutes is expected.

Conclusion

From the results, we can conclude that our baseline CE model shows comparable nowcasting performance with the PySTEPS: Their scores for PCC and FSS indicate their similar maximum skillful lead time and minimum skillful length scale. For categorical metrics, the baseline model shows similar CSI performance as PySTEPS for the 1mm threshold (light rain), but the CSI score becomes worse for the 2mm and 8mm threshold.

The problem could arise from the highly imbalanced dataset, so WCE and EVL are introduced. The results show that both models keep a similar PCC as the baseline while slightly improving the FSS scores. In categorical metrics, the light rain CSI (1mm and 2mm) shows a significant increase but at the cost of increased FAR. So, even though these two methods show an improvement in the overall nowcasting ability compared with baseline, two problems: the poor high threshold CSI score and the increased FAR, remain.

The high precipitation patterns are usually made up of a correct combination of different extreme tokens in the latent space. While EVL does improve the classification between non-extreme and extreme tokens, the classification within the extreme tokens is not considered. The EVL_FA model is introduced for this purpose. The result indicates that it partly solves the previous problem. For light rain, it can keep a similar CSI score as the EVL model while at the same time decreasing the FAR. For a higher threshold, it increases the CSI to a comparable level to PySTEPS. So overall, EVL_FA can be considered the

best-performed model.

5.1.2. Effect of averaging

This section first studies the effect of different averaging numbers on the overall nowcasting performance. The results show the relationship between averaging number and 3-hour average score, and are shown in figure 5.4, 5.5 and 5.6. From left to right, the figures' points indicate different averaging numbers: 1(without averaging), 3, 5, 7, 10, 12, 15, 17, and 20. The results are based on predictions from the model: "EVL_FA", but averaging has a similar effect on other models.

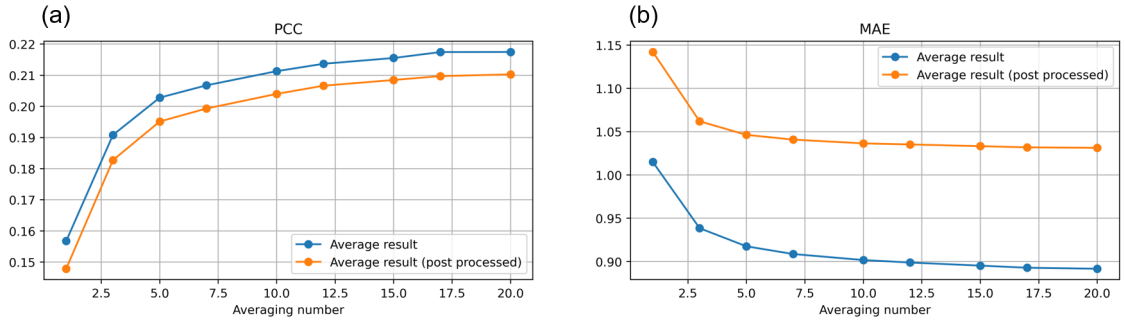

Figure 5.4: Relationship between averaging number and 3-hour averaged PCC (sub-figure a) / MAE (sub-figure b) (Pixel-level evaluation)

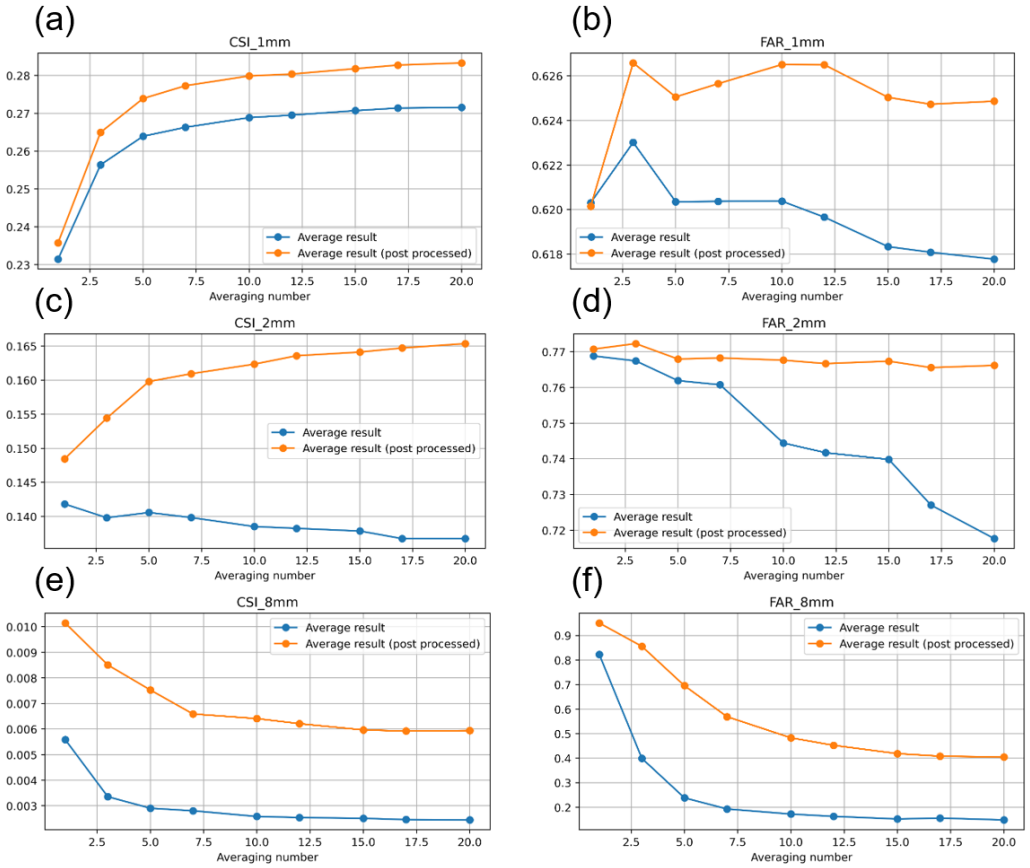

Figure 5.5: Relationship between averaging number and 3-hour averaged CSI (sub-figure a, c, e) / FAR (sub-figure b, d, f) for different thresholds (Pixel-level evaluation)

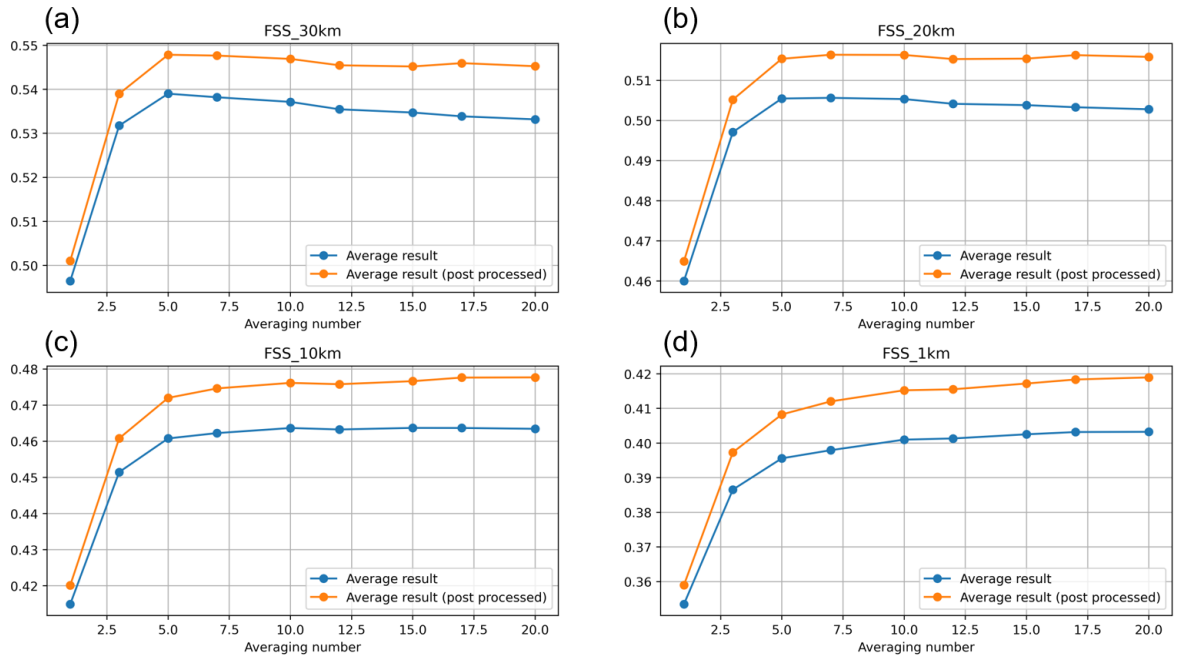

Figure 5.6: Relationship between averaging number and 3-hour averaged FSS for different length scale (Pixel-level evaluation)

As shown from the averaging results (blue curves), compared with the non-averaged results (the curve's first point on the left), averaging improved most metrics, including PCC, MAE, FSS, and FAR. This improvement is also usually more significant with a more averaging number. However, the improvement is less noticeable when the averaging number is more significant than 5 or 7. In terms of CSI, for the 1mm threshold, averaging also brings a better score. However, for the 2mm threshold, increasing the number almost cannot make a difference to the model. For the 8mm threshold, averaging brings a decrease in CSI.

This decrease could probably be explained by the lighter overall rainfall intensity in the averaging results. So, the post-processing method is introduced. As shown from the results (orange curve). Post-processing brings a decrease in PCC, MAE, and FAR performance. Despite the decrease, it still shows better or similar results as the non-averaging model, and FAR is kept in a reasonable range. In addition, it also vastly increases the 8mm threshold CSI and FSS performance.

Based on the performance and considering that a larger averaging number means longer generation time, an averaging number of 5 and the post-processing technique are used in later sections for the extreme event detection task.

5.1.3. Conclusion of nowcasting performance (Pixel-level evaluation)

Metrics/Models	Baseline(CE)	PySTEPs	WCE	EVL
PCC	0.205	0.158	**0.216**	0.202
MAE	**0.802**	0.933	0.922	0.938
CSI(1mm)	0.214	0.210	**0.276**	0.262
CSI(8mm)	0.004	**0.008**	0.004	0.006
FAR(1mm)	0.574	**0.553**	0.605	0.623
FAR(8mm)	**0.318**	0.896	0.423	0.399
FSS(1km)	0.33	0.326	**0.417**	0.394
FSS(10km)	0.404	0.416	**0.5**	0.456
FSS(20km)	0.458	0.473	**0.558**	0.498

Table 5.1: Summary of the 3-hour averaged precipitation nowcasting skill of different models (Pixel-level evaluation)

The table above shows the 3-hour averaged metric values of different models. All deep learning models apply an average number of 5, and PySTEPS applies an ensemble number of 20. The best metric values are in bold, and several conclusions can be reached based on the table:

1. Our baseline CE loss model generally shows a similar level of performance compared with PySTEPs, which proves that our selected model architecture is suitable for the precipitation nowcasting task.
2. By changing class weight (WCE) or adding an additional loss term (EVL), the nowcasting performance shows clear improvement. It overall outperforms PySTEPs, which validates the usefulness of these two methods in precipitation nowcasting tasks. Between WCE and EVL, the WCE shows a slight general advantage over EVL. Before averaging, we can see that EVL actually outperforms WCE in precipitation nowcasting. WCE's higher MAE also indicates that it has more overestimation problems. This problem could arise from the overfitting problem of WCE, which can be alleviated by averaging. So, averaging brings more advantages to the WCE model and leads to a better result.

5.2. Extreme event detection ability

To detect extreme events that happened in the catchment areas, these areas are cropped from the prediction radar precipitation maps produced by different models and evaluated in this section. In this thesis, the extreme events are defined by the catchment averaged precipitation accumulation over three hours, and the extreme thresholds are the top 1% highest average precipitation accumulation (based on data from the MFBS dataset). The thresholds for each catchment are then determined. The table below shows the thresholds of 4 different catchments as an example.

Catchment name	Aa	Delftland	Regge	Dwar
MFBS	8.90mm/3h	10.03mm/3h	8.19mm/3h	8.6mm/3h
RT	4.53mm/3h	4.06mm/3h	4.39mm/3h	4.65mm/3h

Table 5.2: Extreme event thresholds of 4 different catchments

The detection ability is analyzed in two ways: first, the extreme thresholds for catchments are fixed to the determined ones. Then, four metrics: HR, FA, FAR, and CSI are used to evaluate the detection performance for this threshold. Second, a set of different extreme thresholds are assigned to the catchments to evaluate the overall extreme detection ability, and a ROC curve is drawn to compare the models' detection ability. In this case, all catchments are assumed to have the same extreme threshold.

Based on the previous experiment, this experiment uses the forecasting results of different models with five averaging members and the post-processing method.

There are two different configurations for this experiment based on different datasets. The first comparison compares the estimated catchment averaged precipitation accumulation with the ground truth calculated from the RT dataset. While for the second comparison, the CARROTS corrected prediction is compared with the ground truth result estimated from the MFBS dataset. These estimated rainfall accumulation X (in mm/3h) are expressed as equations below.

$$\begin{aligned} X_{pre} &= (X_{pre}^{T+30} + X_{pre}^{T+60} + ... + X_{pre}^{T+180}) * \frac{1}{6} * 3 \\ X_{obs\_RT} &= (X_{obs\_RT}^{T+30} + X_{obs\_RT}^{T+60} + ... + X_{obs\_RT}^{T+180}) * \frac{1}{6} * 3 \\ X_{obs\_MFBS} &= (X_{obs\_MFBS}^{T+5} + X_{obs\_MFBS}^{T+10} + ... + X_{obs\_MFBS}^{T+180}) * \frac{1}{36} * 3 \end{aligned} \tag{5.5}$$

X_{pre}, X_{obs_RT}, and X_{obs_MFBS} represent the precipitation accumulation estimated from the prediction, RT dataset, and MFBS dataset, respectively. X_{pre} and X_{obs_RT} are calculated from the output and reference output of the models. The precipitation is assumed to have little change in the 30-min time interval. While for X_{obs_MFBS}, it is calculated from the more accurate rain-gauge adjusted data, so the estimation can be considered the ground truth accumulation. So overall, from the first comparison, the extreme detection ability difference between different models can be found. While from the second comparison, we can see the models' ability to estimate actual precipitation accumulation.

The data for this experiment contains 357 whole Netherlands events or 3927 catchment-level events, between and 2019-2021, all whole Netherlands events have one or more catchment-level extreme event.

5.2.1. Fixed threshold evaluation

The extreme thresholds for every twelve catchments are set to the top 1% largest catchment average precipitation. (of all rainy events in the catchment during 2008-2014) The results are shown in the two tables below. Four metrics: HR, FA, FAR, and CSI, are used to evaluate the performance (H: True positive, M: False negative, F: False positive, R: True negative). The top two of each metric are bold on the table.

Models/Metrics	HR = H/(H+M)	FA = F/(R+F)	FAR = F/(H+F)	CSI = H/(H+M+F)
PySTEPS	0.5101	**0.1076**	**0.5529**	0.3128
Cross Entropy	0.5235	**0.0721**	**0.4468**	**0.3679**
Weighted CE	**0.7248**	0.1831	0.5970	0.3495
CE + EVL	0.6510	0.2220	0.6667	0.2828
CE + EVL FA	**0.7987**	0.1739	0.5609	**0.3953**

Table 5.3: Summary of the extreme event detection performance of different models (Catchment-level evaluation, RT dataset)

In general, the VQGAN models show better performance than PySTEPS. Compared with PySTEPS, the baseline cross entropy model reaches a bit higher hit rate while having a lower false alarm rate. The weighted cross entropy model successfully increases the hit rate, but the improvement also comes with a larger false alarm ratio. EVL model also shows improvement in detection rate compared with PySTEPS and cross-entropy models, but it shows worse performance than WCE, with higher false alarms but a lower hit rate. While the second version of EVL: the EVL_FA, successfully reaches a balance between hit rate and false alarm rate and achieves the best overall performance (with the largest hit rate while the false alarm rate is kept at a reasonable level).

For MFBS comparison shown in the table below, models show similar performance compared with the previous result: PySTEPS and the cross entropy model still have small FA and FAR but low HR.

The WCE and EVL_FA are the best performing two models in this comparison, but WCE shows a bit better detection ability in this case.

Models/Metrics	HR = H/(H+M)	FA = F/(R+F)	FAR = F/(H+F)	CSI = H/(H+M+F)
PySTEPS	0.3347	**0.0790**	0.4207	0.2692
Cross Entropy	0.3108	**0.0427**	**0.2973**	0.2746
Weighted CE	**0.5219**	0.1218	**0.4178**	**0.3797**
CE + EVL	0.4781	0.1606	0.5082	0.3200
CE + EVL FA	**0.5193**	0.1256	0.4292	**0.3707**

Table 5.4: Summary of the 3-hour extreme event detection performance of different models (Catchment-level evaluation, MFBS dataset)

Conclusion

Based on the above observations, several conclusions can be reached:

1. The proposed EVL and WCE method outperform PySTEPS and the baseline CE model in terms of the extreme event detection rate. This increase in HR also comes with an increase in false alarm rate, which is undesirable. However, the roughly 25% increase in CSI score (compared with cross PySTEPs) indicates the improvement in overall detection ability, proving the effectiveness of applying these new loss functions and the necessity of properly handling the highly unbalanced dataset and the extreme cases.
2. When comparing EVL and WCE, the RT result indicates that the EVL model (CE + EVL 1 + FA) can achieve a higher hit rate while having a lower false alarm rate (and ratio). The weights of WCE are typically inversely proportional to each token's occurrence in the training dataset. This choice is based on the assumption that the data distribution in the training set can reflect the actual distribution. However, the scarcity of extreme samples makes it hard to represent the extreme (right tail) part of the distribution, and the assumption may fail. The EVL essentially also adjusts the weights of the extreme tokens. However, instead of purely relying on the data, the weights are approximated based on EVT by assuming that the area-averaged precipitation accumulation of the extreme tokens follows the heavy-tailed extreme distribution (or Type 2 distribution).
3. The MFBS comparison shows similar relative performance between different models but with a significant decrease in the overall performance. This may be because the MFBS and RT estimation uses different assumptions (RT is based on 30 minutes time interval while MFBS is based on 5 minutes). Also, CARROTS cannot fully compensate for the bias between radar estimated and actual precipitation. Owing to these error factors, the MFBS comparison is unsuitable for comparing different models' performances. However, it provides a helpful reference for how these models perform in predicting actual precipitation accumulation.

5.2.2. Overall extreme detection ability evaluation

To have a better understanding of the models' extreme detection ability at different extreme thresholds, another experiment is conducted. In this case, instead of a fixed threshold for each catchment, a set of different thresholds (different extreme threshold for prediction results but same threshold for the ground truth) are assigned to all catchment in turn. For RT comparison, the common extreme threshold for prediction data is set to 10, 9, 8, 7, 6, 5, 4, 3, 2, 1mm, respectively and the extreme threhold for all catchment are set according to their definition. The ROC curve is shown in figure 5.7.

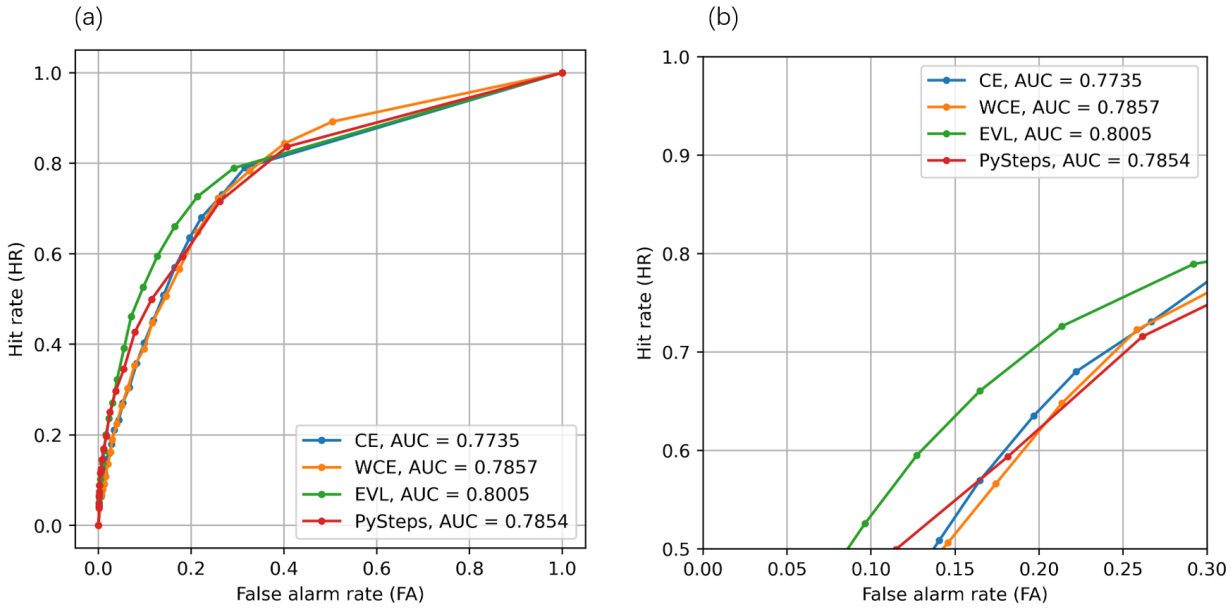

Figure 5.7: (a) The complete ROC curves for 3-hour extreme event detection, the points on the curve (from left to right) represent thresholds of 10mm to 1mm (RT dataset, Catchment-level evaluation). (b) Cropping of the ROC curve by limiting the hit rate to be higher than 0.5 and false alarm rate lower than 0.3

Generally, the results in this section aligned with the results and conclusions from the previous section, the ROC curve shown in figure 5.7 can further prove our conclusions. By comparing different models' area under the curve (AUC), the WCE and EVL both outperforms CE and the WCE achives similar detection performance compared with PySTEPS. Although in terms of the complete curve, the difference between models is not very obvious, if the FA and HR are limited to reasonable ranges (e.g. in our case, HR is limited to 0.5 to 1 and FA is limited to 0 to 0.3 in figure 5.7(b)), the EVL clearly shows a better performance.

6

Conclusion and future work

6.1. Conclusion

This section conclude section concludes the main contributions of this thesis project and our answers to the research questions.

In this thesis work, we propose to use a “VQGAN + Transforme” model for radar extrapolation-based precipitation nowcasting tasks and extreme precipitation event detection. Compared with typical computer vision tasks, one primary difficulty of the nowcasting task is the highly imbalanced distribution of the precipitation intensity. We explore and compare different loss functions (class weight and EVL) to better model extreme events to solve this problem.

Based on the results shown in chapter 5, we can conclude that our proposed model is suitable for the nowcasting task and can show comparable overall nowcasting performance with PySTEPS. Second, the proposed model shows similar extreme event detection ability with the PySTEPS, and the use of EVL vastly improves this detection performance and outperforms PySTEPS.

RQ1: How can we develop a deep generative model that can produce reliable predictions of precipitation field for the following 3 hours?

Applying a deep generative model for precipitation nowcasting tasks is a relatively new research topic. At the start of this project, two relevant models were available from the literature: AENN [16] and DGMR [26]. The main difference is that in Jing [16]’s AENN paper, the generator of AENN applies an encoder-decoder structure for feature extraction and a ConvLSTM-based structure modeling features in the latent space. In contrast, DeepMind’s model [26] down-samples the input radar precipitation map into different scale levels. Then, each down-sampling level’s result is sent to the ConvGRU-based structure for feature extraction and modeling. The two models both apply a combination of temporal and spatial discriminators.

The application of RNN-based models in precipitation nowcasting has been explored for several years since Shi proposed ConvLSTM [31]. In the field of NLP (Nature Language Process), RNN-based models originally played the dominant role because of their sequential nature suit with NLP tasks. However, a new type of deep learning model called Transformer [35] has taken RNN’s place in recent years because of its better performance in capturing long-term dependency. More recently, considerable research has also proven its effectiveness and potential in computer vision tasks. For precipitation nowcasting, it is both a sequential task and a visual-related task. Considering the great success of Transformer in both fields, we plan further to explore its potential in this particular application field.

Based on literature research, the general structure of the model is determined. The structure has two stages: a VQGAN, which can compress the radar data into smaller dimension discrete latent space, in the first stage, and an autoregressive Transformer, which can model this encoded discrete latent

space, in the second stage.

Besides the model, the data is also an important part. The original data is the Netherlands' KNMI archived radar precipitation maps from 2008 to 2020. This thesis work defines an event as a collection of precipitation fields in 3 hours. Since the project focuses on extreme events, we must include as many heavy rainy events as possible in our training set. To achieve such a dataset, first, the precipitation accumulation is estimated for every possible event. No rain events (precipitation accumulation smaller than 0.1mm) are then excluded, and finally, the top 20% of the remaining event are selected to build the training and testing set. The model is then built based on the open source project using PyTorch and trained using the previously selected events.

The result in chapter 5 shows that our trained model can achieve similar performance metric values as PySTEPS, which is considered the STOA nowcasting model. It effectively proves that the proposed model is capable of and suitable for producing skillful prediction of precipitation field.

RQ2: How can we define and detect extreme precipitation events? How can we modify the model to improve the extreme detection ability further?

To define the extreme precipitation event, we first analyzed the statistics of the KNMI radar dataset. Besides analyzing pixels' precipitation intensity, based on [14], we further analyze the statistics of average precipitation accumulation in the catchment areas in the Netherlands. Both of the analysis results indicate that the data distribution is highly unbalanced. More than 90% of the data indicates not rainy (<0.1mm/h for pixel-level). Even when only considering the rainy data, more than 70% of the data indicates light rain (<1mm/h).

The extreme value is defined as values that largely deviate from the median. In this thesis work, we can define the largest 1% of all the rainy values (>0.1mm) as extreme values. So naturally, from the data analysis, the extreme values can be defined from pixel or catchment levels. After careful consideration and discussion with experts, this thesis defines the extremes from the catchment level for several reasons. First, single pixels having extreme precipitation at a single time point may not represent an extreme event in the real world. Instead, only clustering multiple extreme pixels over a certain period may cause an extreme event. Both the spatial and temporal coverage needs to be considered, and the catchment averaged accumulation can meet this requirement. Second, the precipitation intensity is essentially an estimation and thus not accurate. Although this is an unavoidable problem, taking the average over multiple pixels in an area can somewhat compensate for the inaccuracy. So, to conclude, the catchment averaged precipitation accumulation is used to detect the extreme events that happened in catchment areas and the thresholds of each catchment are determined to be the largest 1% of all the catchment average precipitation accumulation.

The highly unbalanced dataset causes the unbalanced distribution of discrete tokens in the latent space of VQGAN (also the input data of the Transformer). The training of the Transformer is essentially solving a multi-class classification problem using cross-entropy loss (to classify the current tokens into one of the 1024 tokens in the codebook). Such imbalance could lead to underfitting and thus poor testing performance on precipitation nowcasting and extreme event detection. Two directions are explored to address this problem: one direction is a classic method called class weight, which typically assigns a higher weight to the class with fewer occurrences. The other direction is to incorporate extreme value theory to better model the data's right tail (or extreme) part.

The weights of WCE are typically inversely proportional to the occurrence of the classes in the training dataset. This is based on the assumption that the provided training data can fully represent the actual distribution, which cannot hold because of the scarcity of extreme data. This problem can be solved by introducing prior extreme value distribution. The data modeled by extreme value distribution needs to be continuous to incorporate extreme value theory. We can set this data as the average precipitation over a specific area based on our definition of the extreme event. This data can be easily connected with our discrete token since one token is encoded from (and can be decoded into) an 16km by 16km area on the precipitation map. All the possible tokens can now be classified into extreme

or non-extreme, and the proposed EVL can be applied by assuming the area averaged precipitation follows a heavy-tailed distribution (and thus, the extreme part follows a type 2 extreme value distribution).

The result in chapter 5 indicates that both WCE and EVL can improve the nowcasting performance over the baseline CE model. While the WCE shows a bit better overall nowcasting skill, the EVL model shows a clear advantage over WCE, CE, and PySTEPS when detecting extreme events that happen in catchments, proving its effectiveness in better modeling our defined extreme event.

6.2. Future work

The section is divided into three parts based on different directions: data, problem formulation, and model.

6.2.1. Data processing

This thesis work focuses on a relatively small area (256*256km) because of the large GPU usage of the proposed model. However, to make a skillful prediction in a longer lead time, the nowcasting task needs to have a larger input area. The input data can be adjusted in two ways: use a radar map with a larger dimension or resize the radar map to the current dimension.

6.2.2. Problem formulation

One difficulty of this project is that there is no clear definition (e.g., a determined threshold) for extreme precipitation events, and we need to define it ourselves. In this thesis work, the extreme event is defined by the three-hour averaged precipitation accumulation in the catchment area. The extreme threshold is set to the largest 1% of all the rainy averaged precipitation accumulation. It is a straightforward definition that makes it easier to prepare data, select events, and build models. However, this definition will only cover part of all extreme precipitation events. For example, an extremely heavy rain only lasts for 20 minutes may not be included in this definition. Another undetectable possible case is that heavy rain happened only in 10% of the catchment area while there is no rain in another part. So overall, the definition is limited. A universal definition and its corresponding model are expected for the future of this project.

The main idea is to increase both temporal and spatial resolution. An example solution is splitting the study area into 5km by 5km grid cells. Mark the cell with ”possible extreme” if the accumulation in the past e.g., 20 minutes, passed the extreme threshold (determined based on each cell’s historical statistics). Then, if clustering of such marked cells is detected in the observation, set an extreme alarm for cities or catchments covered by the cluster.

6.2.3. Model

The proposed model can be further improved in several ways:

1. The current first stage model VQGAN can generate an accurate reconstruction of the original radar map. Moreover, unlike other deep learning nowcasting models, the prediction can keep the same level of sharpness with a longer lead time. However, the reconstruction is still not as detailed and sharp as expected from the original paper. This could be because of the non-optimal tuning of parameters (e.g., the compression rate) or the implementation mismatch. For this thesis work, a large compression rate of 16 and a small codebook size of 1024 are used. From the experiment of the VQGAN paper [10], a such hyper-parameter choice can produce an accurate reconstruction of the precipitation fields. However, the reconstruction result is less sharper than the reconstruction generated by models using a lower compression rate and a larger codebook.
2. There are various improved VQGANs proposed recently worth trying for our task. For example, a much smaller codebook dimension (e.g., 16 instead of 256 or 512 in standard configuration) is proposed to increase the codebook’s usage rate, which is indeed a common problem for the VQGAN used in the thesis. Also, other researchers propose to use $L_2 normalized$ codebook, or equivalently, using cosine similarity to calculate the distance for better convergence speed.

3. For the second stage transformer, the use of sparse attention can speed up the training process and allow the use of a deeper model but also limits the model's function. Specifically, the currently used sparse attention kernel size is $(5, 3, 3)$. In the latent space, it means that the current token attends spatially to the nearby three-by-three area. On the original radar map, it means a 48km by 48km area. If the precipitation field is growing or dissipating very quickly, the change of precipitation field in 30 minutes may need to pay attention to the cells outside the kernel. Furthermore, this kernel choice may face difficulty capturing spatial and temporal features of such change in the precipitation field. This choice is mainly limited by GPU memory and training time. There are several possible solutions for future work: First, the larger kernel size or even full attention can be explored. Second, we can explore a more efficient transformer model (and corresponding attention mechanism). Third, the time interval can be reduced, which is equivalent to applying a larger kernel size.
4. Another disadvantage of our models in terms of the overall nowcasting performance compared with PySTEPs is the low CSI score in case of a high threshold (8mm). CSI is used for pixel-level comparison between prediction and ground truth. So it is safe to assume that such the problem is more related to the VQGAN part than the transformer part, which only handles the area-level discrete tokens. The problem could be that the use of perceptual loss in VQGAN focuses more on the match of the area rather than the match of pixels. To solve the problem, first, we can try to replace the perceptual loss with conventional MSE or MAE or even weighted MSE/MAE (with large precipitation intensity having larger weights), which focuses more on pixel match. Second, this problem could arise from the pixel level data unbalance. Besides applying different class weights, we can also explore different data transformation techniques, such as log transformation.
5. Another drawback of our model is the autoregressive generation process. For comparison, PySTEPS takes around 1 minute to generate six predicted precipitation fields with 20 ensemble numbers, and another deep learning model may only need several seconds. For our autoregressive Transformer, it may take 2-4 minutes to generate six predicted precipitation fields on a GPU. If we apply an average number of 5, it means 10-20 minutes for predicting one event. This long generation time is not avoidable for an autoregressive model. Although Transformer shows excellent performance in many deep learning tasks, it is not the only option to model the latent space prior distribution P(Z). For example, considering that modeling the discrete latent space is similar to a language model, an LSTM structure can be used instead to avoid the long autoregressive generation time. Alternatively, we can explore other commonly used models, such as PixelCNN, for modeling the latent space.

Bibliography

[1] Peiman Asadi, Sebastian Engelke, and Anthony C Davison. “Optimal regionalization of extreme value distributions for flood estimation”. In: *Journal of Hydrology* 556 (2018), pp. 182–193.

[2] H Beekhuis and T Mathijssen. *From pulse to product, highlights of the upgrade project of the Dutch national weather radar network. 10th European Conf. on Radar in Meteorology and Hydrology, Wageningen, Netherlands, Wageningen University and Research, 960–965*. 2018.

[3] AC Best. “The size distribution of raindrops”. In: *Quarterly journal of the royal meteorological society* 76.327 (1950), pp. 16–36.

[4] Younes Boulaguiem et al. “Modeling and simulating spatial extremes by combining extreme value theory with generative adversarial networks”. In: *Environmental Data Science* 1 (2022).

[5] Neill E Bowler, Clive E Pierce, and Alan W Seed. “STEPS: A probabilistic precipitation forecasting scheme which merges an extrapolation nowcast with downscaled NWP”. In: *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography* 132.620 (2006), pp. 2127–2155.

[6] Keith Anthony Browning. “Review lecture: Local weather forecasting”. In: *Proceedings of the royal society of London. A. Mathematical and Physical Sciences* 371.1745 (1980), pp. 179–211.

[7] Guoxing Chen and Wei-Chyung Wang. “Short-Term Precipitation Prediction for Contiguous United States Using Deep Learning”. In: *Geophysical Research Letters* 49.8 (2022). e2022GL097904 2022GL097904, e2022GL097904. DOI: https://doi.org/10.1029/2022GL097904. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2022GL097904. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2022GL097904.

[8] Bin Dai and David Wipf. “Diagnosing and enhancing VAE models”. In: *arXiv preprint arXiv:1903.05789* (2019).

[9] Daizong Ding et al. “Modeling Extreme Events in Time Series Prediction”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*. KDD ’19. Anchorage, AK, USA: Association for Computing Machinery, 2019, pp. 1114–1122. ISBN: 9781450362016. DOI: 10.1145/3292500.3330896. URL: https://doi.org/10.1145/3292500.3330896.

[10] Patrick Esser, Robin Rombach, and Bjorn Ommer. “Taming transformers for high-resolution image synthesis”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 12873–12883.

[11] Janos Galambos. *The asymptotic theory of extreme order statistics*. Tech. rep. 1978.

[12] Urs Germann and Isztar Zawadzki. “Scale-dependence of the predictability of precipitation from continental radar images. Part I: Description of the methodology”. In: *Monthly Weather Review* 130.12 (2002), pp. 2859–2873.

[13] Ian Goodfellow et al. “Generative adversarial networks”. In: *Communications of the ACM* 63.11 (2014), pp. 139–144.

[14] RO Imhoff et al. “Spatial and temporal evaluation of radar rainfall nowcasting techniques on 1,533 events”. In: *Water Resources Research* 56.8 (2020), e2019WR026723.

[15] Ruben Imhoff et al. “A climatological benchmark for operational radar rainfall bias reduction”. In: *Hydrology and Earth System Sciences* 25.7 (2021), pp. 4061–4080.

[16] JR Jing et al. “AENN: A generative adversarial neural network for weather radar echo extrapolation”. In: *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 42 (2019), pp. 89–94.

[17] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. “Perceptual losses for real-time style transfer and super-resolution”. In: *European conference on computer vision*. Springer. 2016, pp. 694–711.

[18] Phuong Dong Le et al. “Dependence properties of spatial rainfall extremes and areal reduction factors”. In: *Journal of Hydrology* 565 (2018), pp. 711–719. ISSN: 0022-1694. DOI: https://doi.org/10.1016/j.jhydrol.2018.08.061. URL: https://www.sciencedirect.com/science/article/pii/S0022169418306644.

[19] Vadim Lebedev et al. “Precipitation nowcasting with satellite imagery”. In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 2680–2688.

[20] Ze Liu et al. “Swin transformer: Hierarchical vision transformer using shifted windows”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10012–10022.

[21] Ze Liu et al. “Video swin transformer”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 3202–3211.

[22] Bruce D Lucas, Takeo Kanade, et al. *An iterative image registration technique with an application to stereo vision*. Vol. 81. Vancouver, 1981.

[23] Rachel Prudden et al. “A review of radar-based nowcasting of precipitation and applicable machine learning techniques”. In: *arXiv preprint arXiv:2005.04988* (2020).

[24] Seppo Pulkkinen et al. “Pysteps: an open-source Python library for probabilistic precipitation nowcasting (v1. 0)”. In: *Geoscientific Model Development* 12.10 (2019), pp. 4185–4219.

[25] Alec Radford et al. “Improving language understanding by generative pre-training”. In: (2018).

[26] Suman Ravuri et al. “Skilful precipitation nowcasting using deep generative models of radar”. In: *Nature* 597.7878 (2021), pp. 672–677.

[27] Nigel M Roberts and Humphrey W Lean. “Scale-selective verification of rainfall accumulations from high-resolution forecasts of convective events”. In: *Monthly Weather Review* 136.1 (2008), pp. 78–97.

[28] Alan W Seed, Clive E Pierce, and Katie Norman. “Formulation and evaluation of a scale decomposition-based stochastic precipitation nowcast scheme”. In: *Water Resources Research* 49.10 (2013), pp. 6624–6641.

[29] AW Seed. “A dynamic and spatial scaling approach to advection forecasting”. In: *Journal of Applied Meteorology* 42.3 (2003), pp. 381–388.

[30] Xingjian Shi and Dit-Yan Yeung. “Machine learning for spatiotemporal sequence forecasting: A survey”. In: *arXiv preprint arXiv:1808.06865* (2018).

[31] Xingjian Shi et al. “Convolutional LSTM network: A machine learning approach for precipitation nowcasting”. In: *Advances in neural information processing systems* 28 (2015).

[32] Xingjian Shi et al. “Deep learning for precipitation nowcasting: A benchmark and a new model”. In: *Advances in neural information processing systems* 30 (2017).

[33] Kevin Trebing, Tomasz Sta□czyk, and Siamak Mehrkanoon. “SmaAt-UNet: Precipitation nowcasting using a small attention-UNet architecture”. In: *Pattern Recognition Letters* 145 (2021), pp. 178–186.

[34] Aaron Van Den Oord, Oriol Vinyals, et al. “Neural discrete representation learning”. In: *Advances in neural information processing systems* 30 (2017).

[35] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[36] Rym Worms. “Propriété asymptotique des excès additifs et valeurs extrêmes: le cas de la loi de Gumbel”. In: *Comptes Rendus de l’Academie des Sciences Series I Mathematics* 5.327 (1998), pp. 509–514.

[37] Chenfei Wu et al. “N\” uwa: Visual synthesis pre-training for neural visual world creation”. In: *arXiv preprint arXiv:2111.12417* (2021).

[38] Wilson Yan et al. “Videogpt: Video generation using vq-vae and transformers”. In: *arXiv preprint arXiv:2104.10157* (2021).

A

Additional experiment result

The first section shows the effect of using different weighting parameters for EVL and the second section shows how averaging affect the nowcasting performance for models trained with different loss functions.

A.1. The effect of EVL weighting parameter

Three different weights are used for EVL: 0.5, 0.75 and 1. Their corresponding nowcsting performance are compared in this section.

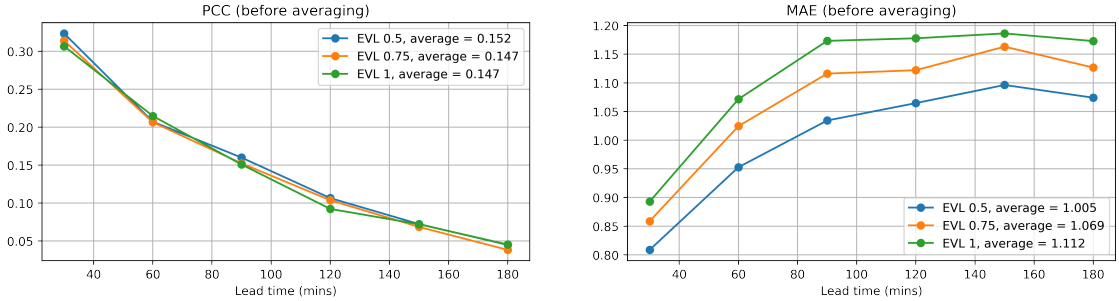

Figure A.1: Comparison of different weights (PCC and MAE)

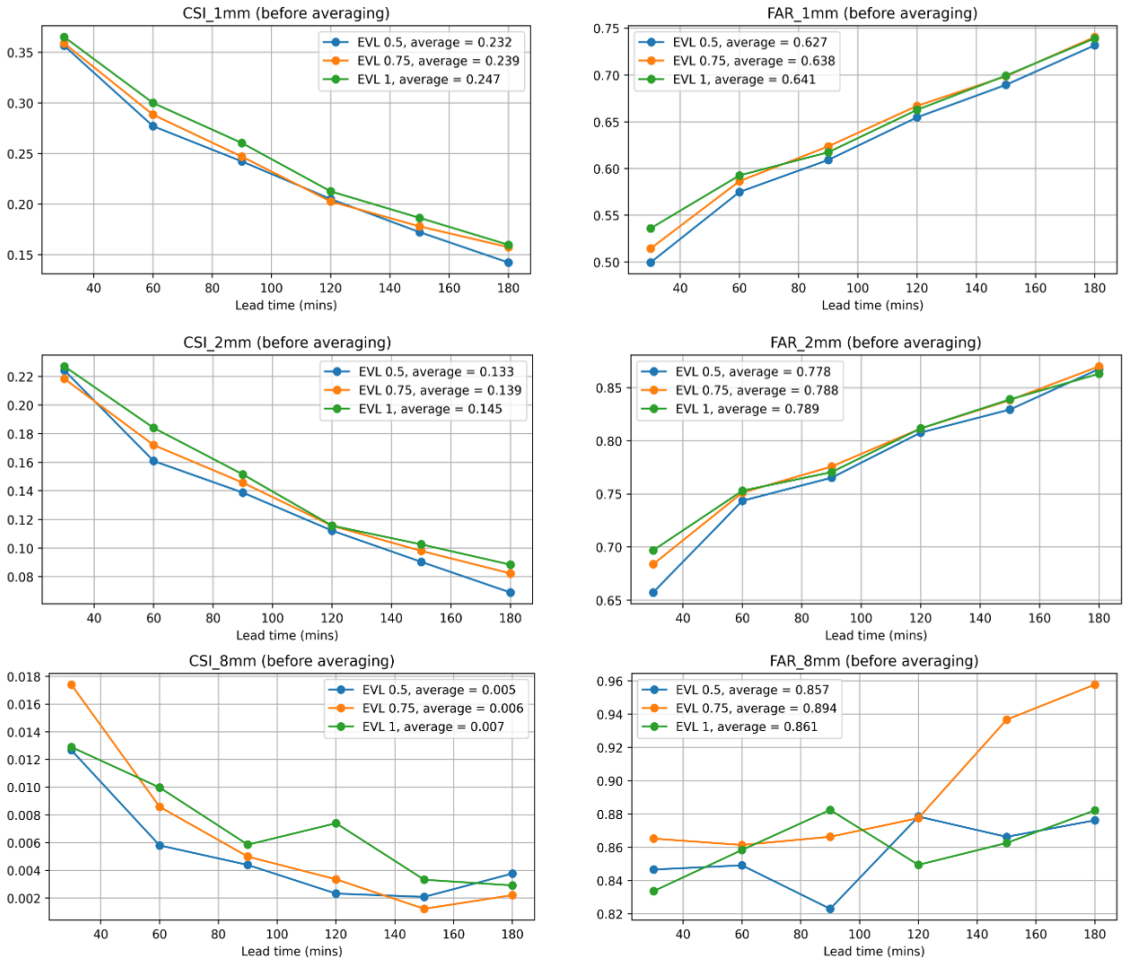

Figure A.2: Comparison of different weights (CSI and FAR)

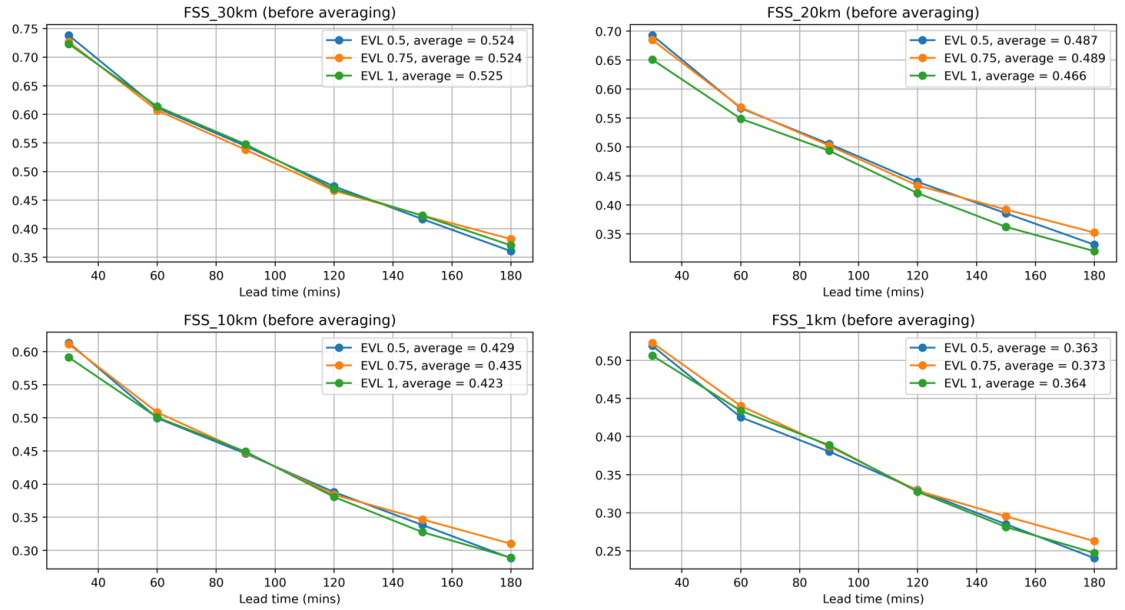

Figure A.3: Comparison of different weights (FSS)

As shown from the results, different weights bring little change to the nowcasting performance in term of PCC and FSS. Increasing weight parameter can bring small improvement in term of CSI with the cost of a bit worse FAR performance. Overall, 1 is used as the default weighting parameter in the experiment.

A.2. The effect of averaging for different loss function

The figures below compares the nowcasting performance before and after averaging. The PySTEPS's values keep the same in the comparison since PySTEPS output is already an ensemble result. An averaging number of 5 is used for this comparison.

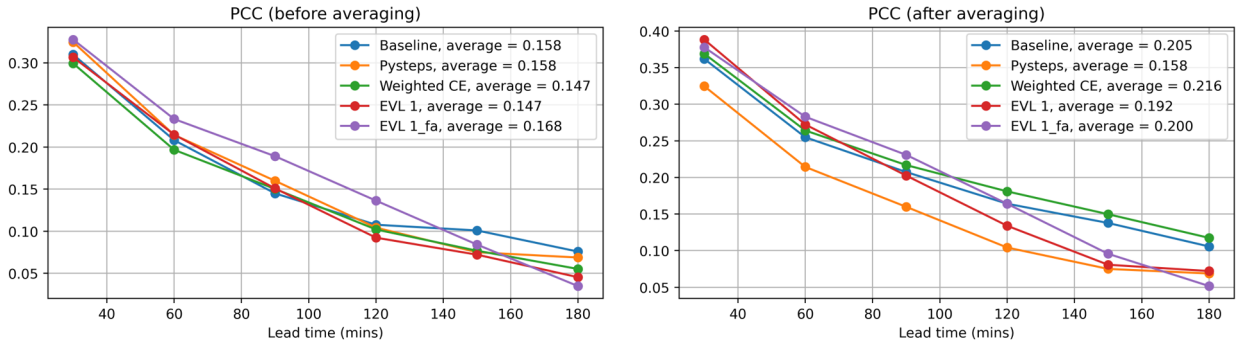

Figure A.4: Comparison of nowcasting performance with/without averaging (PCC)

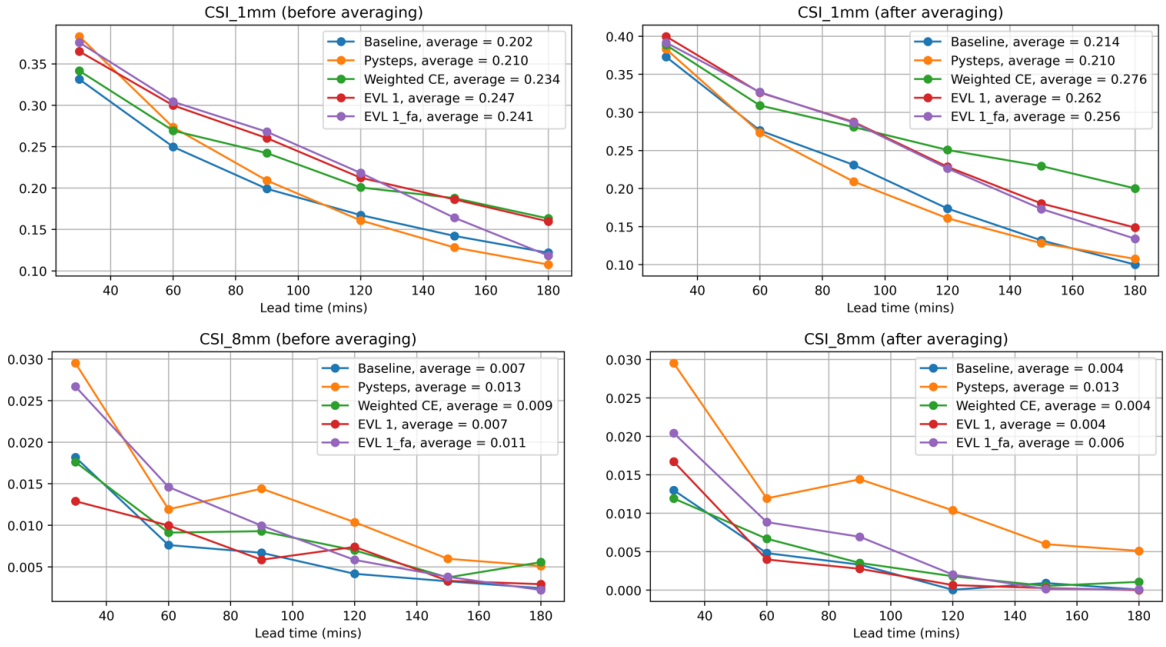

Figure A.5: Comparison of nowcasting performance with/without averaging (CSI)

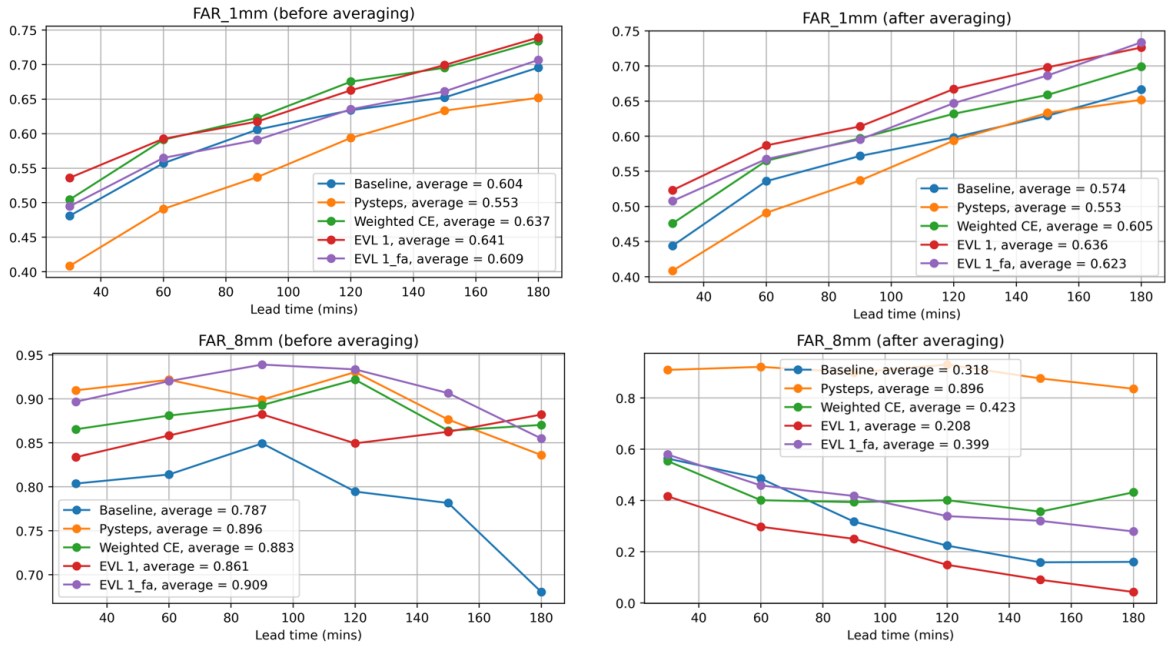

Figure A.6: Comparison of nowcasting performance with/without averaging (FAR)

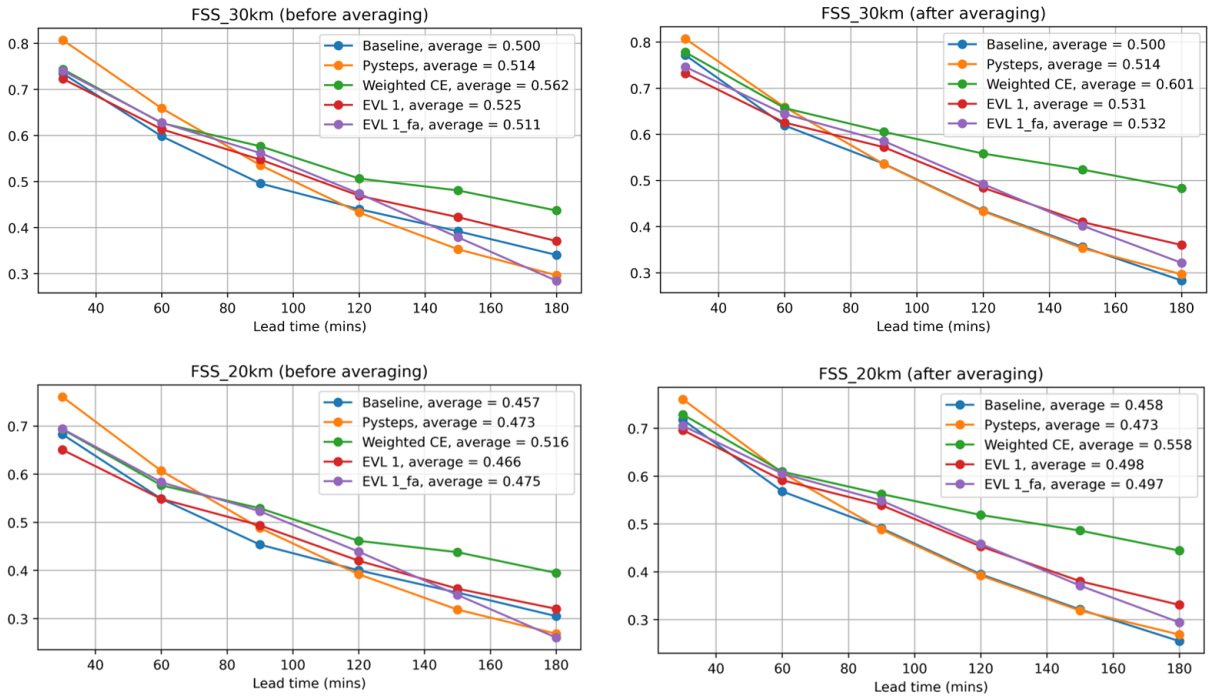

Figure A.7: Comparison of nowcasting performance with/without averaging (FSS)

As already indicated in the chapter 5, averaging make a difference to the nowcasting performance. From figure A.4,A.5,A.6 and A.7, averaging increases the nowcasting performance in term of all the metrics except the 8mm CSI.

B

Examples of nowcasting results

B.1. Example 1

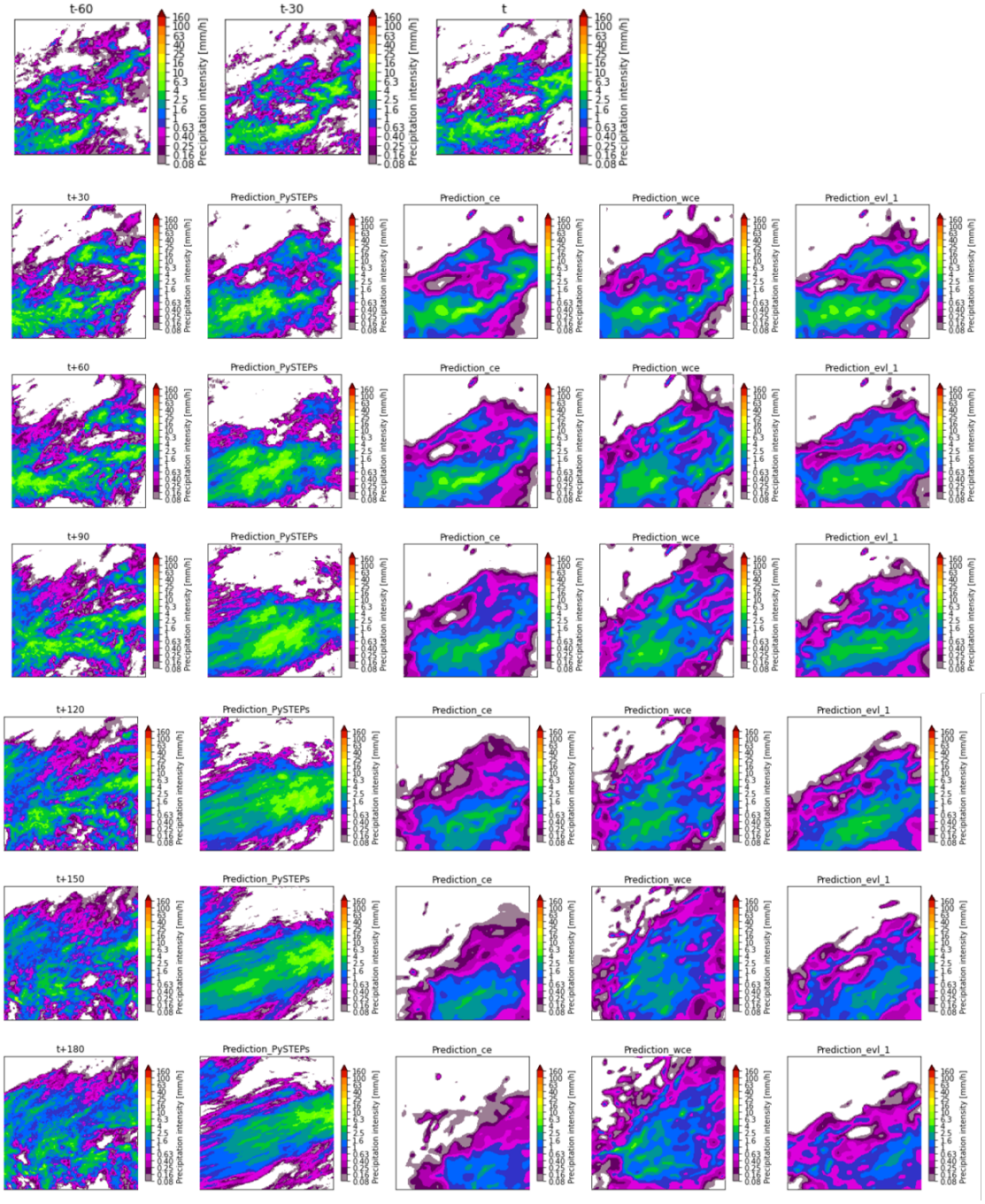

Figure B.1: Comparison of different models' nowcasting result (t = 05:20 2017/07/12)

B.2. Example 2

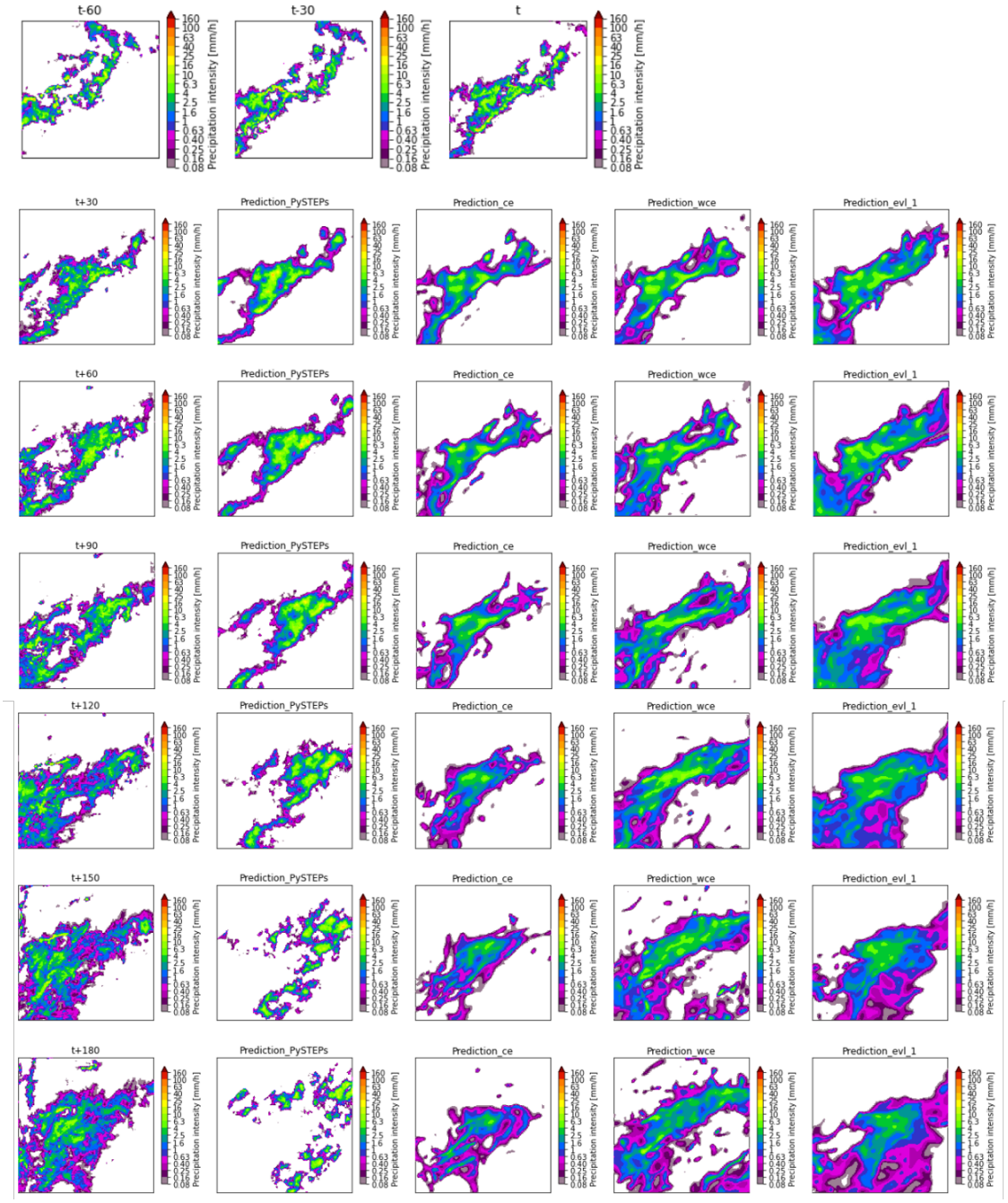

Figure B.2: Comparison of different models' nowcasting result (t = 22:40 2018/08/24)

B.3. Example 3

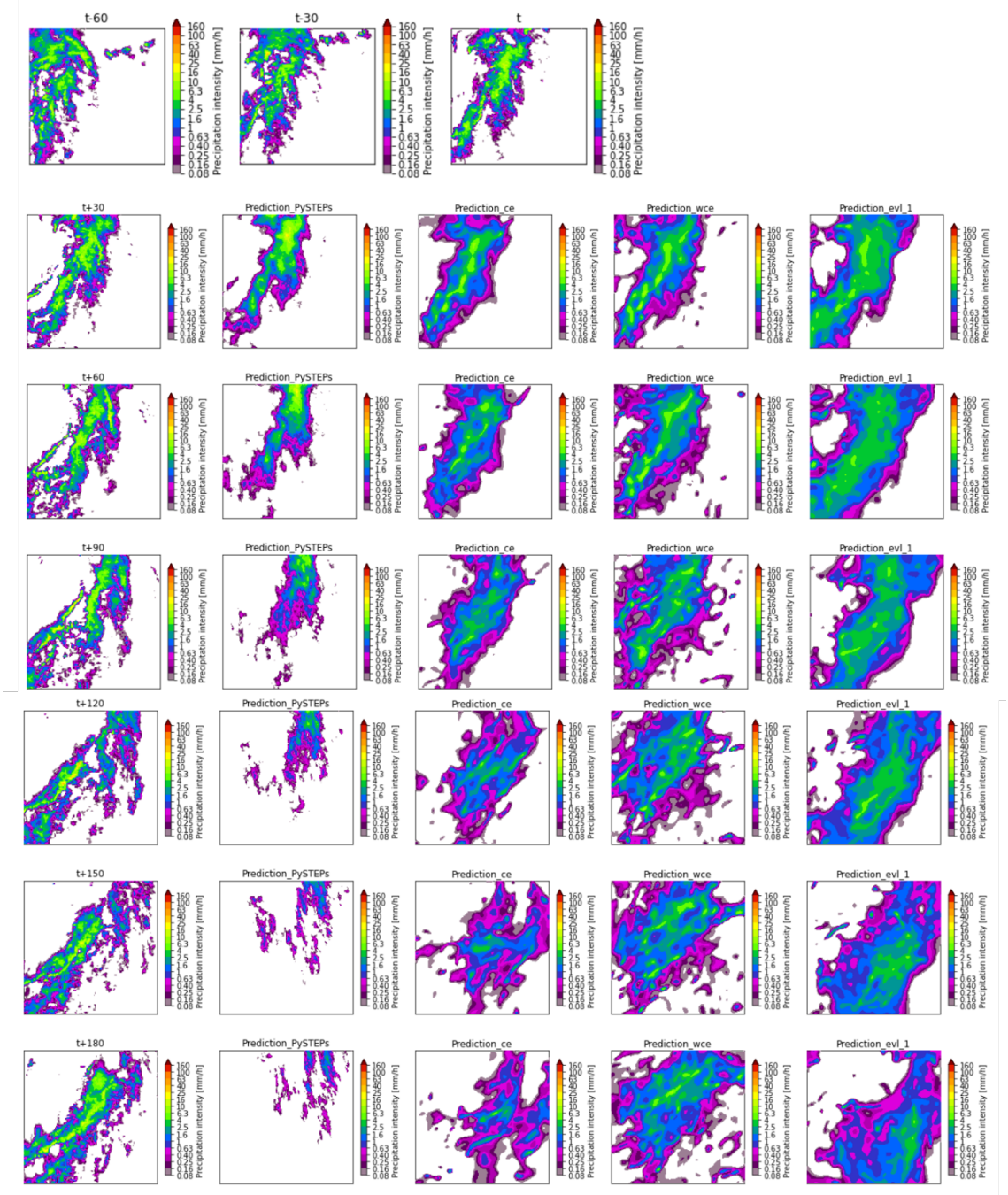

Figure B.3: Comparison of different models' nowcasting result (t = 17:55 2018/08/10)