Master thesis
Data Science

Radboud University

Precipitation Nowcasting using Generative Adversarial Networks

Author:
Koert Schreurs

Supervisor:
Yuliya Shapovalova

External Supervisors (KNMI):
Maurice Schmeits
Kiri Whan

Second Assessor:
Tom Heskes

August 26, 2021

Abstract

Radar extrapolation-based methods and machine learning models typically produce blurry precipitation nowcasts, which are unrealistic. To overcome these limitations, previous research has developed a generative adversarial network to nowcast precipitation (AENN). The AENN combines a reconstruction loss with an adversarial loss. In this thesis we explore the use of GANs for nowcasting precipitation in the Netherlands. We compare the AENN model to the state-of-the-art extrapolation-based model S-PROG. Furthermore, we examine the impact of the adversarial loss by comparing two models with the same architecture, where one receives an adversarial loss and the other does not. Our results show that the adversarial loss reduces the blurriness of the nowcasts and increases the machine learning model's performance for moderate to high rain intensities. Additionally, an extensive evaluation shows that the AENN model performs better at forecasting rain intensities below 2.0 mm/h than S-PROG and S-PROG performs better at predicting moderate to heavy precipitation intensities above 2.0 mm/h. However, none of the models were found to be skillful at predicting moderate to heavy rainfall at a scale of 1 km at a lead time of 30 or more minutes. Further research should be done to improve the performance of the GAN on higher precipitation intensities by using a reconstruction loss that gives more weight to errors in heavy rainfall.

Acknowledgements

I would first like to thank my supervisor from Radboud University, Yuliya Shapovalova, who was always approachable for discussions and provided valuable guidance throughout my thesis. Also, I would like to thank my supervisors from KNMI, Maurice Schmeits and Kiri Whan, for their great support. I really appreciated our weekly discussions, which were really helpful and kept me motivated. Furthermore, I also would like to thank Hidde Leijnse (KNMI) and Aart Overeem (KNMI), for their valuable insights on radar data and clutter. Further, I wish to thank Eva van der Kooij, for her help with the python Pysteps module. Additionally, I would like to thank my parents for their support throughout my study and my uncle, John, for reviewing my thesis on writing errors. Finally, I would like to thank my family, friends and girlfriend for their encouragement and the much needed distractions.

Contents

Chapter 1

Introduction

Rain has a large influence on everyday life. Reliable precipitation forecasts are essential in the decision-making of industries such as agriculture [30], traffic [11], aviation [37] and construction [53]. Precipitation nowcasting, i.e. short-term forecasts of up to 6 hours, are important for water-related risk management. Warning systems rely on nowcasts in order to generate a warning in time for a user to take preventative actions [58, 17]. Given that the frequency and intensity of heavy rainfall events is expected to increase due to climate change, the need for accurate precipitation nowcasting becomes ever more important [3].

Traditionally nowcasts are made using Numerical Weather Prediction (NWP) models or radar extrapolation-based models [5]. NWP models use physics-based models to forecast the weather. These models require a substantial amount of computation power and their spatial and temporal resolution is often too low for precipitation nowcasting. Extrapolation-based models, such as optical flow [10], use observations obtained by weather radars to make forecasts. Optical flow models extrapolate by computing a motion field from past radar data and advecting the most recent radar observations along this trajectory. Radar extrapolation-based models have a higher spatial and temporal resolution than NWP models. Optical-flow methods are designed with the assumptions of Lagrangian persistence in mind, which assumes that 1) the total rain intensity remains constant and 2) the motion field remains constant [19]. However, Lagrangian persistence does not always hold as the motion and intensity can vary with respect to time and thus these assumptions can hinder the model's performance.

Recent studies have successfully applied machine learning (ML) for weather nowcasting [66, 54, 61]. Machine learning methods, such as deep learning (DL), take advantage of the large amount of available historical weather data. These models are not hindered by the assumption of Lagrangian persistence. However, the deep learning approaches tend to result in predictions that are smooth/blurry and look unrealistic as result of optimizing with a standard loss function like mean-squared-error (MSE) [60]. A class of generative ML frameworks called Generative Adversarial Network (GAN) has been to shown to address this issue [26, 33, 59]. Jing et al. proposed the Adversarial Extrapolation Neural Network (AENN) GAN model that outperformed other state-of-the-art models [33]. In the GAN framework two DL models are pitted against each other in an adversarial way. The generator generates synthetic samples and the discriminator has to discriminate between real and synthetic samples. The generative model does not train to minimize the distance to a target image, but it trains to fool

the discriminator. Ideally, this results in the generating learning to approximate the target distribution, resulting in realistic forecasts.

In this thesis, we explore the use of GANs to perform precipitation nowcasting over the Netherlands. The GAN model AENN was trained and evaluated on Dutch precipitation radar data made available by the Royal Dutch Meteorological Institute (KNMI). Additionally, we included a more extensive verification compared to previous studies. The GAN model was compared to the state-of-the-art optical flow model Spectral Prognosis (S-PROG) [52]. Furthermore, we demonstrated the influence of an adversarial loss by comparing a generator trained with adversarial loss and without it.

The remaining part of this thesis is organized as follows. In Chapter 2 further background is given on machine learning in weather prediction. Chapter 3 gives details about the dataset used and explains the methodology that was used to select samples from the dataset. Chapter 4 provides details about the GAN model that was used in this study. Chapter 5 explains the models that were used as a baseline for the GAN model. Chapter 6 discusses the evaluation metrics that were used. The results are presented in Chapter 7 and discussed in Chapter 8. Chapter 9 discusses the conclusions.

Chapter 2

Related Work

In this Chapter we present a brief overview of related work on machine learning in weather forecasting.

Radar extrapolation is a sequence prediction problem. Therefore, it is closely linked to recurrent neural networks like long short-term memory (LSTM) [27] and gated recurrent unit (GRU) [13]. Additionally, radar echo maps can be represented as images making the task of radar extrapolation also a computer vision problem. Therefore, the task is also linked to convolutional neural networks (CNN) that can deal with spatial structures in the data.

Sutskever et al. proposed a general sequence-to-sequence LSTM framework for sequence prediction [56]. Shi et al. extended this framework to deal with the radar extrapolation problem[66]. The authors proposed a novel convolutional LSTM (ConvLSTM) model for precipitation nowcasting. The sequence-to-sequence framework was extended by using multiple ConvLSTM layers to create an encoding-forecasting framework. The ConvLSTM model provides a framework for dealing with spatiotemporal relationships. The Critical Success Index (CSI), False Alarm Rate (FAR), Probability of Detection (POD), and correlation scores of predicting 0.5 mm/h rain rate threshold was used to evaluate the model. The authors show that their model outperforms the state-of-the-art optical flow model ROVER. In a follow up paper, Shi et al. showed that other categories of RNN, like GRU, can also be extended into a ConvRNN framework [54]. Furthermore, the authors proposed the Trajectory GRU (TrajGRU) model, which improves upon ConvRNN models. Unlike the ConvRNN models, The TrajGRU can learn location-variant structure. Additionally, they introduced Balanced Mean Squared Error (B-MSE) loss function. The B-MSE loss function penalizes errors on heavy precipitation more, resulting in better performance on heavier precipitation. Recent work by Van der Kooij assessed the performance of the TrajGRU model in the Netherlands [61]. The author showed that the TrajGRU was able to outperform the optical-flow method S-PROG.

A common downside of using deep learning approaches to forecast precipitation is that the results look blurry and are not realistic. GAN have seen great success in generating realistic synthetics images [21]. GANs have also been used for weather forecasting. Bihlo used an ensemble of GANs for forecasting different weather variables [9]. The predicted variables were the geopotential height of the 500hPa pressure level, the two-meter temperature and the total precipitation for the next 24 hours in Europe. The GAN model was able to reach a good per-

formance on all variables except for total precipitation. Furthermore, Dai used an ensemble of GANs for post-processing NWP forecasts of cloud-cover [16]. The author showed that the GAN model outperformed traditional state-of-the-art port-processing models. Furthermore, the GAN's output was less blurry and looked more realistic when compared to other deep learning approaches.

Some notable work has also been done on nowcasting precipitation using GANs. Hayatbini et al. proposed a conditional GAN to estimate precipitation [26]. Two models were developed for this task. They first trained a model to predict a rain/no-rain binary map. Then the regression model predicts for each rainy pixel its quantity. The latter was done with a GAN network. The authors used the U-Net architecture [50] for the generator model. The regression model uses a combination of MSE loss and adversarial loss. The proposed model was shown to outperform other baseline models that were trained without an adversarial loss component. Furthermore, Tian et al. propose a GAN model (GA-ConvGRU) where the generator uses a ConvGRU framework and the discriminator is a CNN [59]. The generator was trained with a loss function that combines MSE and the adversarial loss. The authors used two years for training data and excluded images with an average rainfall rate below 0.01 mm/h. The authors evaluate their model by using the metrics POD, FAR, CSI and Heidke Skill Score (HSS) [65, Chapter 8.2] in combination with thresholds of 0.5, 5, 10, and 30 mm/h. The GA-ConvGRU was shown to obtain a better POD score than ConvGRU across all thresholds, however for the other metrics the two models performed similarly. Additionally, the authors showed that their GAN model yield more realistic results when compared to the ConvGRU framework. Besides Jing et al. proposed the Adversarial Extrapolation Neural Network (AENN) model, which is a GAN framework with one generator and two discriminators [33]. This model improves upon the ConvLSTM framework by adding an adversarial loss to the generator. The generator creates an echo frame sequence predicting 0.5, 1 and 1.5 hours ahead. The ConvLSTM framework was used for the generator part of the network. Furthermore, the AENN model consists of two discriminators. One looks at individual frames and the other looks at the sequence. For the verification metrics the authors used the POD, FAR, CSI and HSS scores with a threshold of 0.5 mm/h. The AENN model was compared with a state-of-the-art optical flow model and with the ConvLSTM model. The AENN model was able to obtain the best POD, CSI and HSS score, however the ConvLSTM was able to achieve a better FAR score. The authors showed that the output of AENN is more realistic and less blurry than the output of the ConvLSTM model.

Chapter 3

Data

The precipitation data that was used was collected by KNMI. KNMI has two radars, located in Herwijnen and Den Helder, which measure radar reflectively in dBZ every five minutes. The radar product estimates rainfall accumulation over 5 minutes by converting the dBZ values to rain intensity by using the Z-R relationship [41]: $Z = 200R^{1.6}$, where Z is the radar reflectivity factor in dB and R is the rainfall rate in mm/h. The two radars cover the Netherlands plus areas over sea and across the Belgian and German borders.

The real-time radar product [8] is available every 5 minutes. A mean field bias correction is applied to this data by using automatic rain gauges. From this point on we will call this dataset the real-time (RT) dataset. An example of a few radar scans from this dataset can be seen in Figure 3.1.

The images in the RT dataset have an image size of 765 by 700 pixels. Each pixel represents a grid of 1 km by 1 km. The radar images are made at an altitude of 1.5 km. The accumulation values were converted to rain rates (mm/h) by multiplying the rain accumulation in 5 minutes by a factor of 12. The values were clipped at a maximum of 100 mm/h. Values above this are likely to be caused by hail or by echoes from sources other than precipitation (i.e., clutter). This reduces the impact of false extremes, while still allowing to predict high precipitation values. Additionally, we proposed a method to reduce clutter in

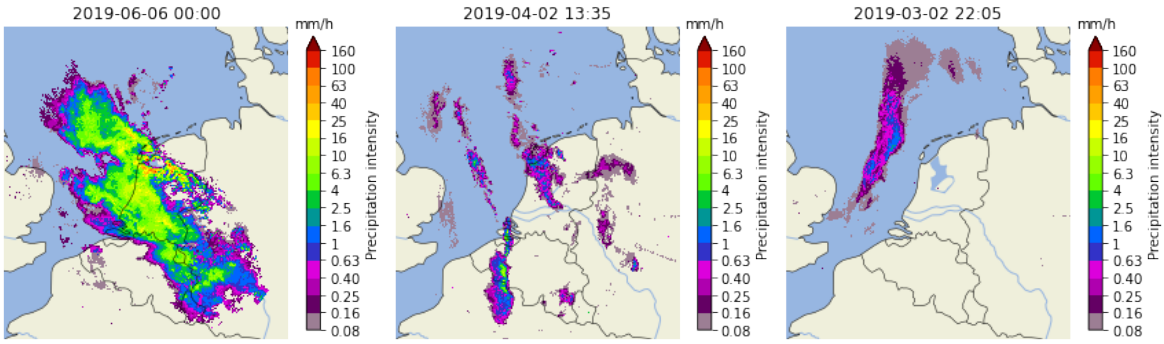


Figure 3.1: Examples from the real-time radar product. The same colormap is used throughout the thesis for precipitation radar images.

the dataset, which is described in section 3.1. This method discards radar scans that have high probability of containing clutter.

Furthermore, we also discarded samples that had very low mean amounts of precipitation. Most of the time it does not rain in the Netherlands. Therefore, a large part of the data consists of samples without rain. The focus of this thesis lies on forecasting precipitation. Therefore, a set of rainy events was sampled from the real-time data set. We define a rainy event as 6 consecutive 5 minute samples with rain and a radar scan was labeled as rainy if the average rain intensity per pixel exceeds 0.01 mm/h. A total of 41% of the data was labeled as containing rain. Of all 30 minute slices in the data 38% were rainy events. Figure 3.2 shows the percentage of rainy samples per month. The pixel statistics of the rainy samples can be seen in Figure 3.4b.

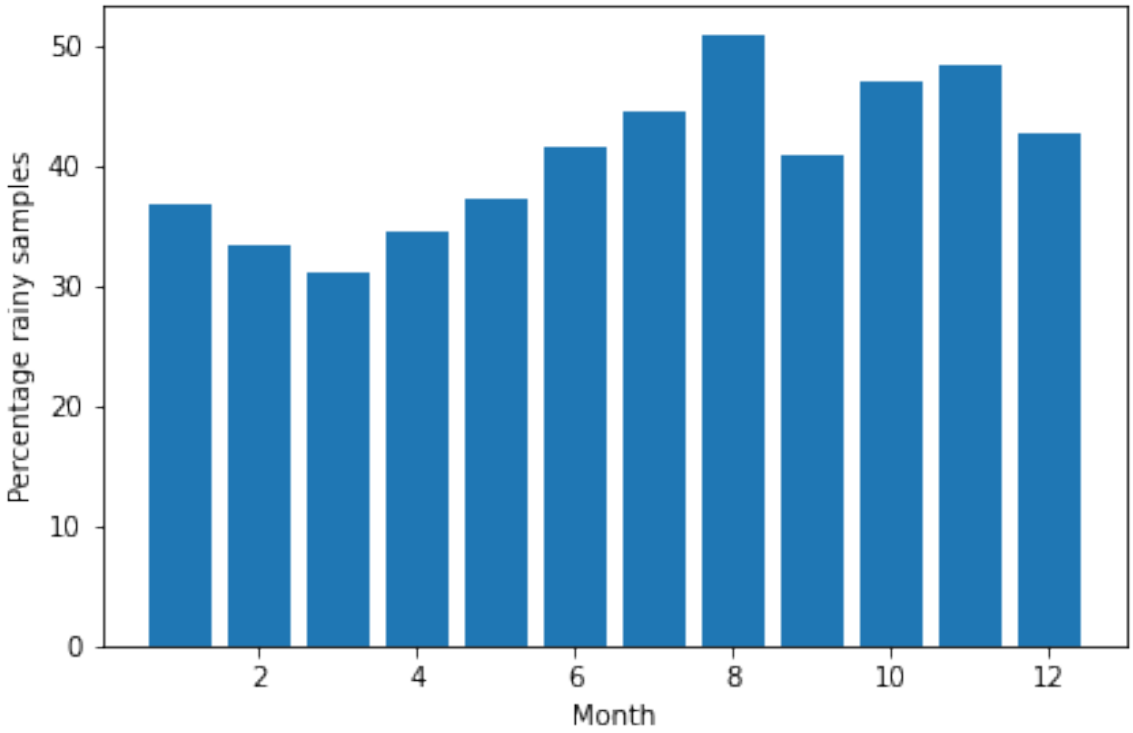


Figure 3.2: Percentage of rainy samples per month

3.1 Clutter

In this section we give some background what clutter is and why it is unwanted. Furthermore, we will briefly discuss the different clutter removal methods that were explored and the reason why they were not used. Lastly we introduce a method that discards samples if they are likely to contain clutter.

Unwanted echoes can be generated by objects that are not the target of the radar. These

echoes are called clutter. In this project any echoes generated by things other than precipitation are seen as clutter. Clutter can be caused by echoes returned from the various sources like the ground (ground clutter), the sea (sea clutter), airplanes, birds/insects, wind turbines or ships. Clutter can have a negative impact on the performance of the model. Furthermore, having a lot of clutter in the dataset would give an unfair advantage to a machine learning method as such method is able to learn to deal with clutter while an extrapolation method cannot do this. Additionally, the clutter results in the radar overestimating the total amount of precipitation, which could be misleading when labeling samples as rainy or non-rainy.

Multiple preprocessing methods are carried out by KNMI on the raw radar data to reduce the clutter before the final radar product is realized [64, 63, 28]. However, there is still clutter in the real-time radar product. Sea clutter and moving objects like ships and wind turbine remain hard to filter out. In Figure 3.3 you can see artifacts in the sea that are caused by clutter. The figure also shows that the clutter artifacts can be relatively large, spanning multiple pixels.

A wide range of methods can be used to remove the clutter from the precipitation radar data itself. One of the simplest methods is to create a static clutter mask that discards pixels that have high probability to contain clutter [34, 15, 54]. Other methods rely on the fact that rain and clutter have a different spatial decorrelation time, in order to distinguish ground clutter [2, 55]. Other studies have used data from sources other than radar, such as satellite images [12, 32] and predictions from a NWP model [7, 6]. Further studies have applied machine learning to distinguish precipitation from clutter [31, 24]. It was beyond this research's scope to perform an extensive removal of clutter in the real-time radar product. Nonetheless, we explored the use of different methods for the removal of clutter, such as applying clutter masks, removing small-scale structures, and discarding pixels with high gradients. However, removing the clutter per sample was a challenging task and no adequate simple solution for this problem was found. In the end we did not apply any clutter removal to our dataset directly, but instead we proposed a method to discard samples that are likely to contain clutter.

In order to see which areas of the radar scan tend to contain clutter, we can look at the statistics per pixel in the dataset. In Figure 3.4a the probability of a pixel exceeding a rain intensity threshold is shown. Places that tend to contain clutter are more likely to exceed the rain thresholds. In Figure 3.4 you can see certain spots in the sea (sea clutter) and ship tracks that often contain clutter.

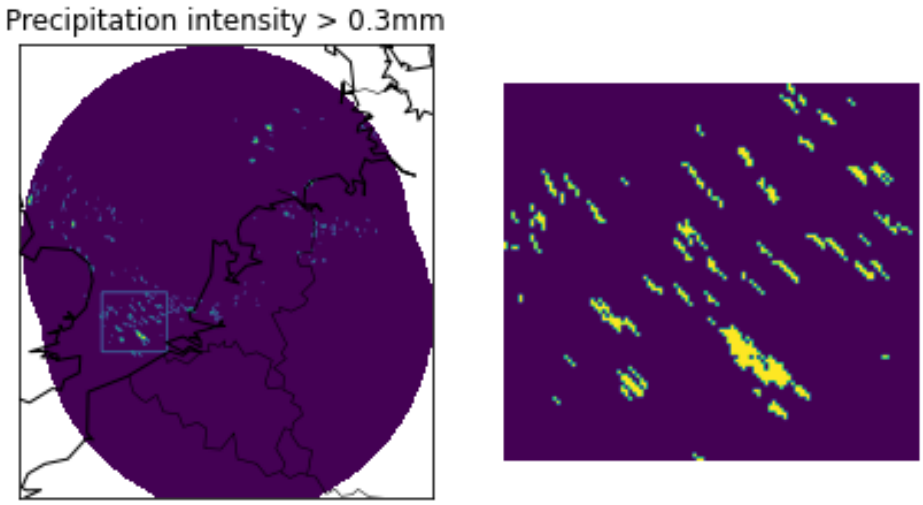
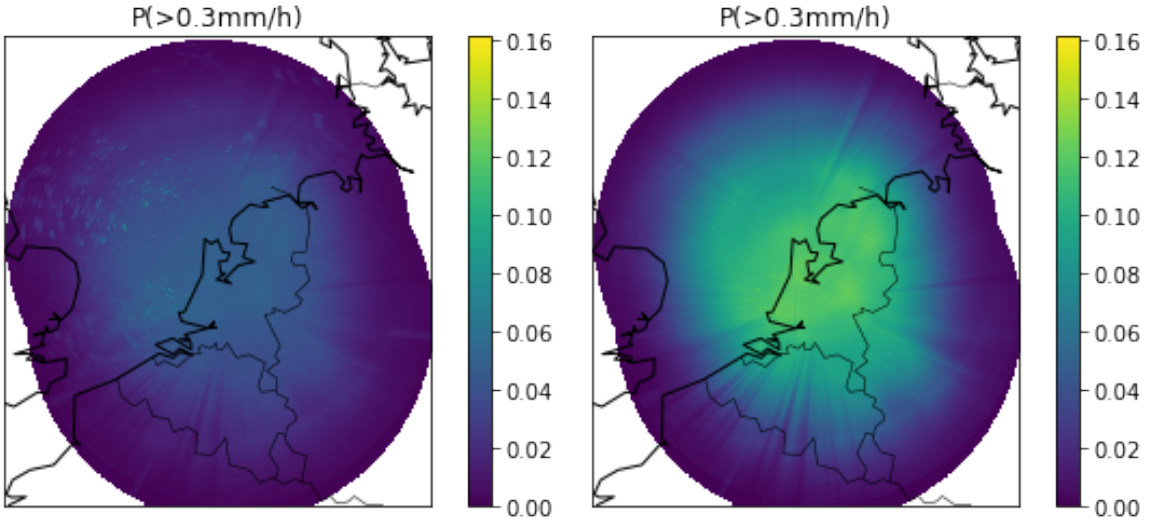


Figure 3.3: This figure shows pixels with higher precipitation intensity than 0.3 mm/h. The left image shows the complete radar scan with a box drawn around an area with a lot of clutter. On the left we zoomed in on this area of the image. Artifacts are visible in the radar data caused by clutter.



(a) (b)

Figure 3.4: Probability of pixels exceeding 0.3 mm/h of rain (a) in the complete dataset, (b) in samples that are labeled as rainy and non-clutter

One possible method of removing clutter is to discard patches of non-zero pixels below a certain size. Clutter artifacts often consist of only a few pixels while precipitation tends to span larger areas. However, this method risks removing small local showers. We define a object as a cluster of 8-connected[1] non-zero pixels. We can then remove objects with a size smaller than n. Choosing the right value of n is important. If n is too small, clutter is not filtered out and if n is too big, real precipitation will be removed from the radar image. Figure 3.5 shows what happens if we remove objects that span less than 15 pixels. A lot of the sea clutter is removed, however removing an area of 15 km^2 can also remove a lot of small scale precipitation. Additionally, it did not remove all clutter in the images. Therefore, we decided not to use this method of clutter removal.

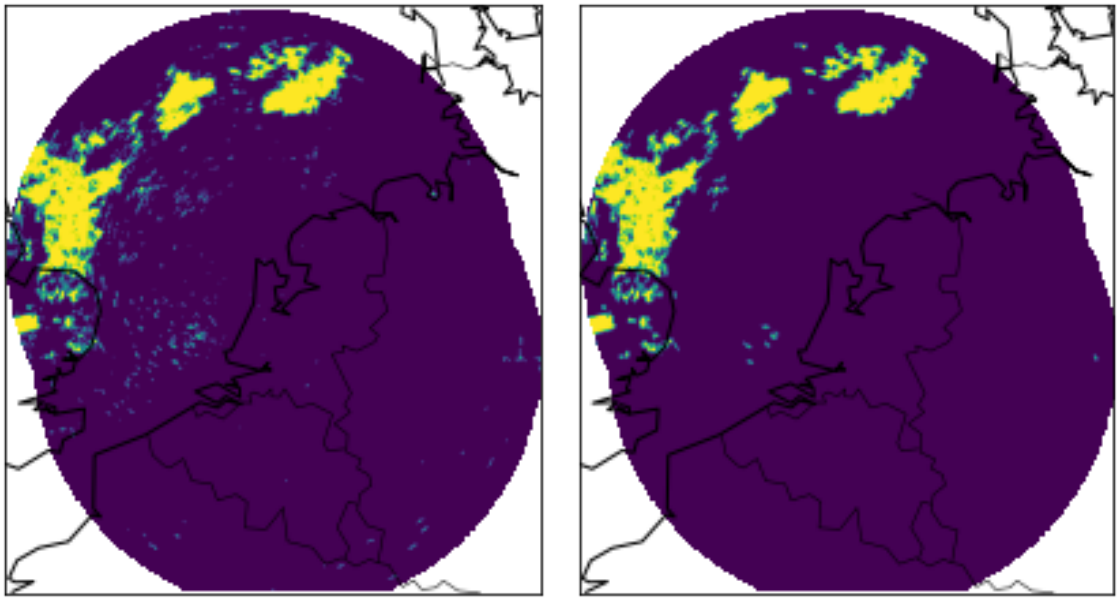


Figure 3.5: Example of clutter removal by removing objects smaller than 15 pixels. Non-zero pixels are in yellow.

Another method to remove clutter is by eliminating pixels that have a high probability of containing clutter. The pixel statistics reveal certain areas in the image that have a high probability of containing clutter (see Figure 3.4a) By using these pixel statistics a static clutter mask can be made. However, downside of using a clutter mask is that it does not filter out all the clutter. Not all clutter is static and can be filtered out by a clutter mask. Furthermore, the clutter mask will also filter out real precipitation. For these reasons we did not use a clutter mask in this thesis.

[1] 8-connected means a pixel has 8 neighbouring pixels. Two pixels are connected if they neighbour each other in the horizontal, vertical or diagonal direction

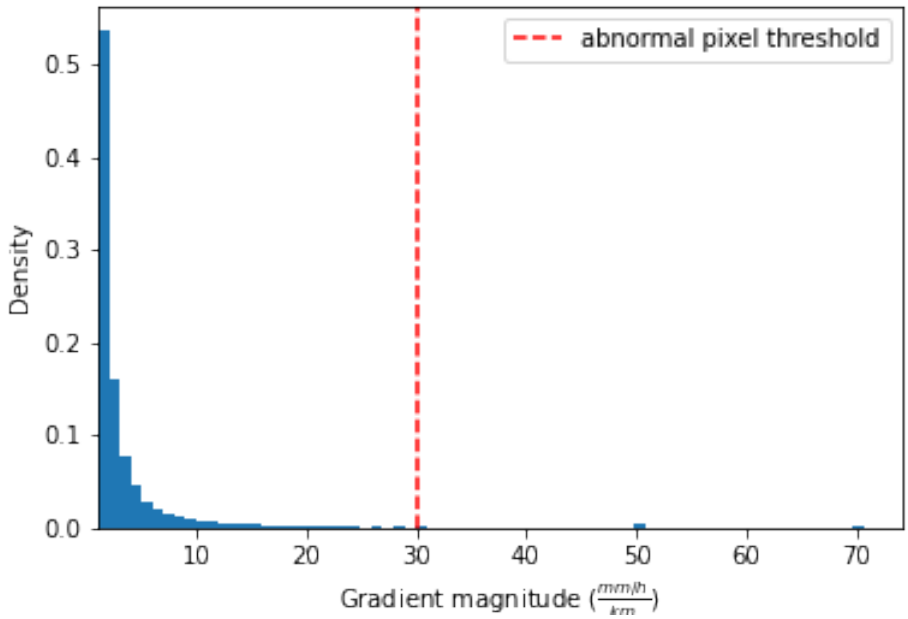


Figure 3.6: Histogram of gradient magnitudes higher than 1 mm/h/km. A threshold was set at a gradient magnitude of $30\frac{mm/h}{km}$. Pixels above this threshold are seen as abnormal

Another approach that was explored was to detect clutter by looking at the gradient of the pixels in the radar image. Rainfall tends to gradually increase from low to high precipitation values, while clutter can cause sudden increases and decreases in radar reflectively, which results in abnormally large gradients. However, discarding pixels based on their gradient can lead to problems when a cloud moves through areas that contain clutter or occasionally precipitation also contains high gradients. This makes it difficult to discard pixels based on the magnitude of their gradient. Instead we propose to use the number of high gradients in a radar image as an indication of the amount of clutter in the radar scan. Images with a lot of clutter can be discarded. The magnitude of the gradients was calculated by computing the gradient from left to right and top to bottom of the image and then taking the square root of sum of squares. A histogram of gradient magnitudes is depicted in Figure 3.6. A threshold was set at a gradient magnitude of 30 mm/h/km. Pixels with a gradient bigger than this threshold are seen as abnormal. Samples with clutter can be discarded by setting another threshold for the number of abnormal pixels allowed in the image and discarding radar scans that exceed this threshold. It was empirically found that images with more than 50 abnormal pixels tend to contain clutter. Therefore, we discarded samples from the dataset if they have more than 50 abnormal pixels. Random samples of radar scans with more and fewer than 50 abnormal pixels can be seen in Appendix A.1 and A.2.

In conclusion, we reduced the amount of clutter in our dataset by discarding samples that contain many pixels with a high gradient. We showed that such samples often contain clutter. No further preprocessing of the data was done to remove clutter in the samples.

Chapter 4

Generative Adversarial Networks

4.1 Background

In the Generative adversarial Networks (GANs) framework two networks compete against each other [21]. The generator creates synthetic samples and the discriminator predicts if its input comes from the training set or from the generator. The generator tries to fool the discriminator into thinking its generated samples are real. On the other hand the goal of the discriminator is to detect fake samples. The arms race between the two models leads to a continuous improvement of both the discriminator and the generator. When one gets better at its task, the other has to improve its game. Ideally after finishing training the generator has learned to approximate the target distribution. The discriminator can then no longer differentiate between the synthetic and the real samples. GANs have seen a broad range of applications such as text-to-image generation [67], generating artificial human faces [35] and image super-resolution [38]. Furthermore, GANs have recently been used successfully in meteorology to forecast cloud cover [9, 16] and precipitation [26, 33, 59].

In this thesis the AENN model was trained and validated on the real-time radar product. The model receives an input of 30 minutes, consisting of 6 samples. The model is tasked with forecasting the precipitation at lead times of 30, 60 and 90 minutes. Additionally, we also explored the use of machine learning for forecasting radar data that was further bias-corrected, but we failed to get reasonable results (see Appendix A.4). Given the time constraints of this research, we instead focused on forecasting the real-time radar product.

4.2 Loss

The mechanisms of the GAN framework proposed by Goodfellow et al. can be described in the following way [21]. Let D be the discriminator and G be the generator. The discriminator D receives an input tensor and classifies whether it comes from the real distribution p_r or from the generator p_g. The generator G receives random noise $z \sim p_z(z)$ as input and finds a mapping that maximizes the probability of D labeling the generated image $G(z)$ as real, $\mathbb{E}_{z\sim p_z}[\log D(G(z))]$. The objective of the discriminator is to maximize its output on the real samples x, $\mathbb{E}_{x\sim p_r}[\log D(x)]$ and minimize its output for fake samples, $\mathbb{E}_{z\sim p_z}[\log D(G(z))]$. Minimizing $\mathbb{E}_{z\sim p_z}[\log D(G(z))]$ is the same as maximizing $\mathbb{E}_{z\sim p_z}[\log((1 - D(G(z))]$. By combining these formula the GANs loss function can be written as a minimax game between G

and D:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_r}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))] \tag{4.1}$$

GANs are capable of generating random realistic samples from a given dataset [21]. However, in this framework there is no control over the images that are generated.
In order to make a proper forecast, the GAN output should be dependent on recent observations. In a conditional GAN (cGAN) this control is achieved by conditioning G and D on another variable c [21, 43]. The conditional value function of the cGAN can be defined as:

$$\min_G \max_D V_c(D,G) = \mathbb{E}_{x\sim p_r}[\log D(x|c)] + \mathbb{E}[\log(1 - D(G(c)))] \tag{4.2}$$

Another method to make the GAN's output conditional is to give the generator the additional task to minimize the distance between $G(c)$ and the target value y. This results in the output having to match two criteria: 1) the nowcast should match the ground-truth as close as possible, this is measured with the reconstruction loss (e.g. MSE or MAE). 2) the nowcast should also fool the discriminator into thinking the generated samples are real, this is measured with the adversarial loss. Recent work on radar extrapolation using GANs have used this approach either in combination with a cGAN [26, 33] or with just a GAN model [59]. As reconstruction loss, l_{rec}, Jing et al. used the sum of the MSE and the MAE for the AENN's generator model [33].

$$\text{MSE} = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{3}\sum_{i=1}^{765}\sum_{j=1}^{700}(y_{n,t}(i,j) - x_{n,t}(i,j))^2 \tag{4.3}$$

$$\text{MAE} = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{3}\sum_{i=1}^{765}\sum_{j=1}^{700}|y_{n,t}(i,j) - x_{n,t}(i,j)| \tag{4.4}$$

where N is the number of samples, t is the lead time, y_n is the target value of sample n, x_n is the predicted value of n and i and j are the row and column index of the pixel. It was empirically found that the model was more stable when only using the MSE as reconstruction loss, $\mathcal{L}_{rec} = \mathcal{L}_{mse}$. We speculate that the reason for this is that the MAE gives relatively more weight to small mistakes. The radar scans in general contain a lot of pixels with no rain, these pixels have a value of exactly 0. Being slightly off on a lot of these pixels significantly lowers the MAE. The MSE is less sensitive to this. The AENN generator uses a ReLU activation [44] in its final layer that sets negative values to 0. A strategy to get a low MAE would be to make sure the network's output is always negative. Then the ReLU activation causes the output of the network to become exactly 0, resulting in a lower MAE score. However, the ReLU has gradient of 0 for negative values. Therefore, the model cannot use gradient descent to get out of this local minimum and gets stuck during training.

In the AENN model there are two discriminators:

1. The frame discriminator, D_{fra}, judges each individual frame independently. The objective function of the frame discriminator, $\mathcal{L}_{fra}$, is to maximize for time t the sum over $n \in (6, 12, 18)$[1] in Eq. 4.1, with $x = x_{t+n}$ and G receives as input $c = (x_{t-5}, x_{t-4}, x_{t-3}, x_{t-2}, x_{t-1}, x_t)$ instead of noise variable z.

[1]Note that the interval between samples is 5 min., so the lead times of 30, 60 and 90 min. correspond to the samples $t + 6$, $t + 12$ and $t + 18$, respectively

2. The sequence discriminator, D_{seq}, is a cGAN and receives as input the sequence of observations and either the target values or the predictions. The objective of the sequence discriminator, $\mathcal{L}_{seq}$, is to maximize Eq. 4.2 with $c = (x_{t-5}, x_{t-4}, x_{t-3}, x_{t-2}, x_{t-1}, x_t)$ and x $= (x_{t+6}, x_{t+12}, x_{t+18})$

The generator tries to minimize the objective function of both discriminators. The complete value function of the AENN model as used in this paper can be described as follows:

$$\min_{G} \max_{D_{fra}, D_{seq}} V_{aenn}(D_{fra}, D_{seq}, G) = \lambda_{adv}\mathcal{L}_{fra} + \lambda_{adv}\mathcal{L}_{seq} + \lambda_{rec}\mathcal{L}_{mse} \tag{4.5}$$

where λ_{adv}, λ_{rec} are the weights for the adversarial loss and the reconstruction loss, respectively.

4.3 Preprocessing

The AENN model was designed to work with input and output image dimensions of 256 by 256 pixels. To comply with this, preprocessing was done to downscale the dataset to 256 by 256 pixels. In our original datasets the images have a dimension of 765 by 700. For the real time dataset zero padding was used to make the images a square of 768 by 768. Afterward, bilinear interpolation [18] was applied in order to reduce the image size by a factor of three, making its size 256 by 256 pixels. As in the original AENN paper, we converted the rainfall rate values to dBZ values using the Z-R relationship($Z = 200R^{1.6}$) [41]. Furthermore, min-max normalization was used to normalize the values between 0 and 1. The preprocessing steps were undone during the validation and testing of the model.

4.4 Generator

The generator receives an input sequence of past radar observations and outputs another sequence of radar images that make up the forecast of the model. This task is called sequence to sequence prediction. The network uses an encoder-decoder architecture [56, 66]. The spatial features of the input sequences are first encoded into a feature space by using multiple convolution layers. These layers are strided causing the dimensionality of the input to be reduced after each layer. This encourages the model to learn a compact representation of the input data. The sequence of encoded images is then processed by Convolution Recurrent Neural Network (ConvRNN) layers. These layers can deal with both spatial and temporal patterns in the data. The ConvRNN forecasts encoded images. These encoded images are then decoded by the decoder part of the network that consists of strided transposed convolutional layers which upsample the output to the desired dimensions. In Figure 4.1 the architecture of the generator is depicted.

The encoder consists of 3 convolutional layers. These layers have 32, 64 and 128 filters, respectively. All layers have a stride of 2 by 2, such that after each layer the image dimensions are halved. The first layer has a kernel size of 5 by 5 and the other layers have a kernel size of 3 by 3. In the original AENN model an ConvLSTM was used to deal with temporal-spatial dependencies. In this project we explore the use of ConvGRU instead of a ConvLSTM for this layer. Two ConvGRU layers are used, the first one encodes the spatial-temporal sequence into a feature space. The second ConvGRU unrolls this feature space into

a prediction. Both layers have 128 filters and a kernel size of 3 by 3. The decoder part of the network is the encoder in reverse. It uses 3 transposed convolutional layers with 64, 32, 1 filters, respectively. All layers have a stride of 2 and a kernel size of 3. The last layer uses a Rectified Linear Unit (ReLU) activation ($f(x) = \max(0, x)$) [44]. This prevents the model from predicting negative precipitation levels. The other layers use a leaky ReLU activation, $f(x) = \max(\alpha x, x)$, where the slope α was set to 0.2 [40]. The slope prevents the problem of a zero-gradient that is present in the ReLU activation function. A zero-gradient can be problematic as no gradient-based optimization can be performed to adjust the neurons weights. This can result in a neuron becoming inactive forever because it is only able to update its weights when it is active.

4.5 Discriminator

The AENN uses two discriminators. One only looks at a single frame and the other looks at the complete sequence. The generator tries to create both a realistic image as well as a realistic sequence. The two discriminators share the same network architecture. The architecture of the discriminator is visualized in Figure 4.2.

The discriminator consists of 5 convolutional layers. These layers have 32, 64, 128, 256 and 512 filters, respectively. A kernel size of 3 by 3 is used except for the first layer which has a kernel size of 5 by 5. Each layer has a stride of 2 by 2. The last convolutional layer is followed by an average pooling layer which is then followed by a dense layer with 1 output. The last layer uses a sigmoid activation function. The other layers use a leaky ReLU activation with a slope of 0.2. The frame discriminator receives an input of length 1 is provided, which is either the prediction or the target at a lead time 30, 60 or 90 minutes. The sequence discriminator receives an input of length 9, which consists of half an hour of observations together with the forecasts or targets at the three lead times.

4.6 Hyperparameter Tuning

Various tests were done in order to find optimal parameter settings for the AENN model. Because of time constraints no extensive hyperparameter search was done, therefore the final hyperparameter setting is likely not the most optimal. We iteratively changed hyperparameters of the AENN model to see if it improved the model's results on the validation set. The hyperparameter setting that obtained the highest performance on the validation set was kept.

Neural network classifiers can be prone to producing extremely confident predictions, especially when the input is constructed in an adversarial manner [22]. This is not good for regularization. A technique called label smoothing can be used to counteract this. Label smoothing has been shown to improve stability and training of GAN models [51, 57]. We can apply label smoothing to the GAN by providing the discriminator with smoothed targets such as 0.1 and 0.9 instead of 0 and 1. The target label y can be smoothed with the formula:

$$y * (1.0 - \alpha) + 0.5 * \alpha \tag{4.6}$$

where α controls the amount of smoothing, when α is 0 no smoothing occurs. We compared the effect of label smoothing ($\alpha = 0.2$) versus no label smoothing ($\alpha = 0$). It was found that

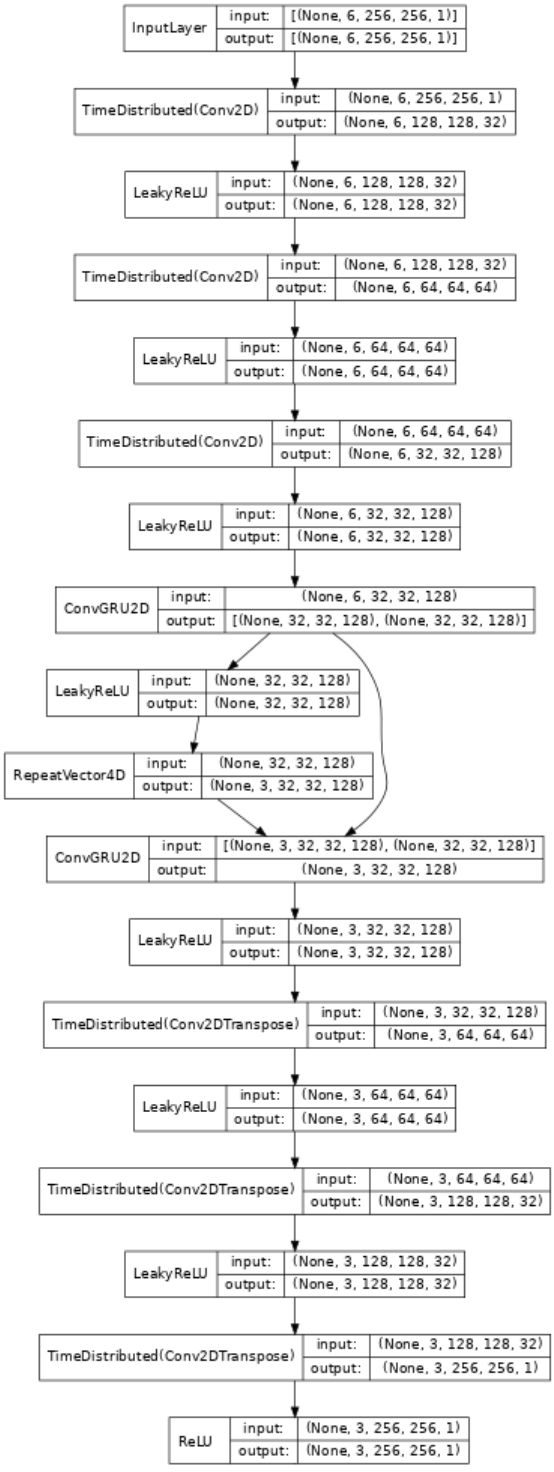


Figure 4.1: Architecture of the AENN-GRU generator. The batch size is indicated by None. The ConvGRU2D can be changed to ConvLSTM2D to obtain the original AENN settings.

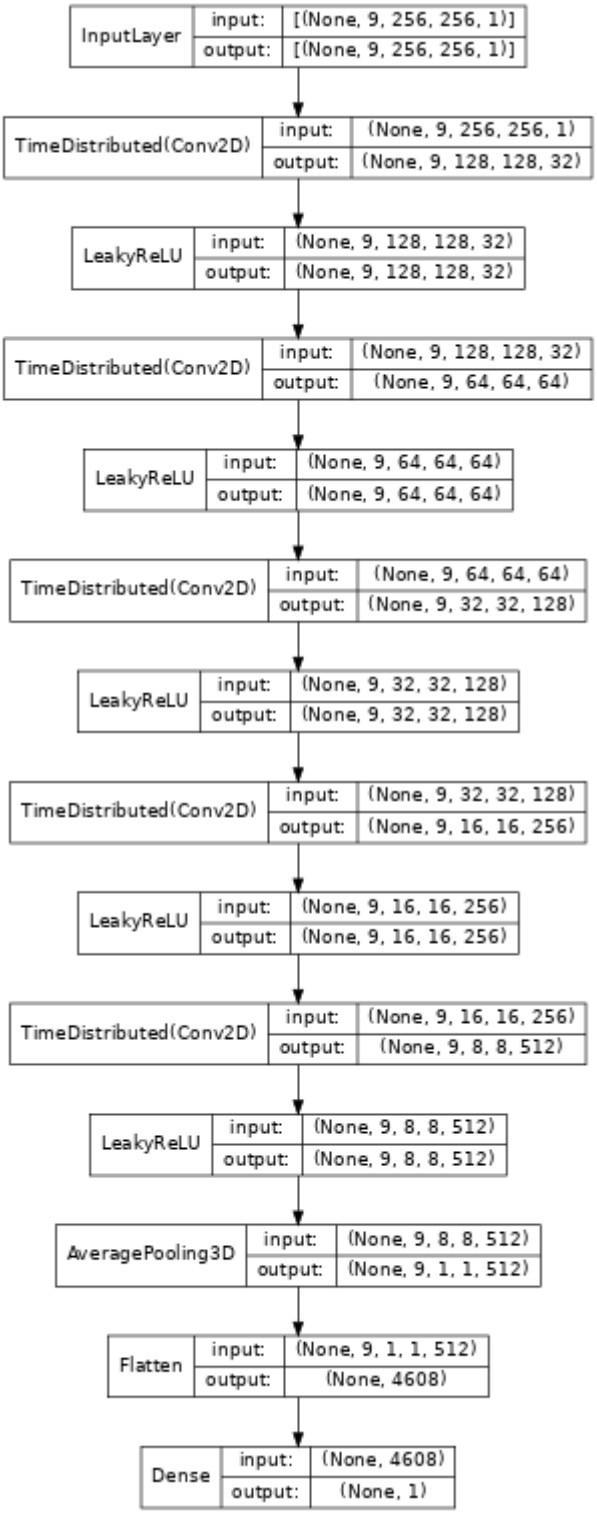


Figure 4.2: Architecture of the sequence discriminator. The frame discriminator has the same architecture except the input sequence length is 1 instead of 9. The batch size is indicated by None.

	CSI					MSE		
	Threshold (mm/h)					Lead times(min.)		
	0	**0.5**	**5**	**10**	**30**	**30**	**60**	**90**
S-PROG	<u>0.487</u>	<u>0.346</u>	**0.094**	**0.033**	**0.001**	0.182	0.223	0.264
AENN	0.461	**0.353**	0.047	0.010	0.000	**0.143**	**0.170**	**0.195**
AENN-ConvGRU	**0.491**	0.343	<u>0.070</u>	<u>0.016</u>	0.000	<u>0.150</u>	<u>0.190</u>	<u>0.213</u>

Table 4.1: Comparison between AENN, AENN-ConvGRU and the S-PROG baseline in terms of CSI across all lead times per rain threshold in mm/h (higher is better) and MSE per lead time in minutes (lower is better). The best result is indicated in bold and the second best result is underlined.

smoothing the labels of the discriminator slightly improved the GAN's validation performance (see Appendix A.3) . Therefore, from this point on label smoothing with an α of 0.2 was applied to the AENN model.

Additionally, we compared the performance of the AENN model with different ConvRNN layers. The AENN model uses ConvLSTM for the ConvRNN layers. However, recent studies have explored the use of ConvGRU layers in precipitation nowcasting [54, 59]. We compared the AENN model with ConvLSTM layers (AENN) versus the AENN model with ConvGRU layers (AENN-ConvGRU). In Table 4.1 a comparison between AENN and AENN-ConvGRU and the baseline S-PROG is depicted. The CSI scores were computed across all lead times. It was found that the AENN had a lower MSE on the validation dataset. However, the CSI scores reveal that the AENN-ConvGRU is better at predicting precipitation of intensities above 5.0 mm/h. Given that correctly predicting higher levels of precipitation is more important to us, we opted to use the AENN-ConvGRU model from this point on.

Lastly we experimented with using different learning rates for the generator and discriminator of the AENN model. The learning rates that were considered were: 1×10^{-4}, 5×10^{-5} and 1×10^{-5} . It was found that lowering the learning rate did not improve performance of the AENN model (see Appendix A.4). Therefore, we kept the learning rate at its original value of 1×10^{-4}.

4.7 Experimental Settings

The model was implemented in python using the Tensorflow Keras library [1]. The code is available on Github[2]. The weights of the neurons were set by using the Glorot uniform initialization [20] with 0 bias. The weight of the reconstruction loss, λ_{rec}, was set to 1 and the weight of the adversarial loss, λ_{adv}, was set to 0.003 as proposed in the original AENN paper. Both the generator and the discriminators use an Adam optimizer [36] with learning rate of 0.0001. ConvGRU was used for the ConvRNN part of the AENN model. Furthermore, the generator was trained 3 times for every update step of the discriminator. The best model in terms of validation reconstruction loss was selected after 100 epochs with early stopping after 20 epochs of seeing no improvements, in terms of validation reconstruction loss. The batch

[2]https://github.com/KoertS/precipitation-nowcasting-using-GANs

size was set to 16, which was the largest possible value that could fit in memory. From now on, we will simply refer to this model with these settings as the GAN model.

Chapter 5

Baseline

The performance of the GAN was compared with two other models. The radar extrapolation-based model S-PROG [23] serves as a baseline for the GAN. Furthermore, we compared the GAN generator with a generator that was trained without adversarial loss, in order to assess the improvements that the adversarial loss provides.

5.1 Radar Extrapolation

Extrapolation-based models make predictions by moving the precipitation along its most recent direction with its most recent speed. The trajectory is derived from recent observations. Unlike the GAN model these models do not need training. In order to make an extrapolation-based forecast the following two steps need to be performed:

1. First a motion field has to be computed based upon recent observations.
2. Secondly, the motion field is used to move the precipitation along its current trajectory.

These steps come from the assumption of Lagrangian persistence [19]. This assumption entails that the precipitation will move with the same speed in the same direction over the course of the forecasting period.

5.1.1 Spectral Prognosis (S-PROG)

Research has shown that the lifetime of precipitation is dependent on its spatial scale [62, 23]. Large-scale rain features generally have a longer lifetime than small-scale rain features. Seed proposed the Spectral Prognosis (S-PROG) model [52]. This extrapolation-based model handles different scales of precipitation in different ways. S-PROG decomposes the precipitation field into a multiplicative cascade. The different levels of the cascade represent different scale rain features.

An implementation of the S-PROG algorithm is available in the Python library [47]. Pysteps is an open-source library for precipitation nowcasting. Imhoff et al. compared multiple open-source precipitation nowcasting algorithms with each other [29]. The algorithms that were compared are: Rainymotion Sparse, Rainymotion DenseRotation, Pysteps deterministic (S-PROG) and Pysteps probabilistic (STEPS) with 20 ensemble members. The authors found that Pysteps deterministic had the longest average decorrelation times. Furthermore, they

showed that in general the two Pysteps algorithms outperform the Rainymotion algorithms. The authors speculate that most errors come from the algorithms not taking into account the growth and dissipation processes. In this study we used the same S-PROG setup as in Imhoff et al.. The setup consists of the following steps. First, the rain rates were converted to dBZ values. After that, the motion field was determined by using a dense Lucas-Kanade optical flow method [39]. Semi-Lagrangian backward extrapolation was applied to the reflectivity field. The order of the autoregressive model was set to 2 and eight cascade levels were used. With a probability matching method, the statistics of the forecast are matched with those of the observations based on the mean of the observations. Lastly the forecast was transformed back from dBZ to rain rate (mm/h).

5.2 Non-adversarial Generator (NAG)

The adversarial loss forces the GAN model to make predictions that look like the real data. If we leave this adversarial loss, this restriction is left out. To examine the influence of the adversarial loss, we implemented a non-adversarial generator (NAG). The NAG shares the exact same architecture as the GAN's generator. Furthermore, we used the exact same settings for the NAG model as for the GAN model. The only difference is that the NAG's loss function consists of only the MSE component without any adversarial loss. The weight of the adversarial loss, λ_{adv}, was set to 0. The expectation is that this model will outperform the GAN model in terms of MSE, as its only objective is to minimize this score. However, we also expect that the predictions of NAG will look smoother and more unrealistic than the GAN's predictions because the NAG does not have an adversarial loss.

Chapter 6

Evaluation

Before we can compare models to the target values, we need to apply some post processing to the machine learning models' output. The predictions are denormalized and then converted back from dBZ to rainfall rate by using the Z-R relationship ($Z = 200R^{1.6}$) [41]. We evaluate the model on the original image with size of 765 by 700. The model output's is of size 256 by 256, so in order to compute the metrics we need to upsample the output. We used bilinear interpolation to upsample the model's output to a size of 768 by 768 (factor of 3). Then we applied cropping to get to the dimensions of 765 by 700.

A number of different metrics were used in order to evaluate the model and the baseline methods. The metrics can be divided into continuous scores and categorical scores.

6.1 Continuous Scores

The two continuous score metrics that were considered are the Mean Squared Error (MSE) and the Mean Absolute Error (MAE). The metrics are calculated for the lead times 30, 60 and 90 minutes. The MSE and MAE are defined as follows:

$$\text{MSE} = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{765}\sum_{j=1}^{700}(y_n(i,j) - x_n(i,j))^2 \tag{6.1}$$

$$\text{MAE} = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{765}\sum_{j=1}^{700}|y_n(i,j) - x_n(i,j)| \tag{6.2}$$

where N is the number of samples, y_n is the target value of sample n, x_n is the predicted value of n and i and j are the row and column index of the pixel.

6.2 Categorical Scores

The categorical scores show the performance of the model on different levels of rain intensity. The scores can be computed by converting the target and the predictions to binary images by setting a rain threshold. To see the model performance on different rain intensities, we used 7 different rainfall rate thresholds. The thresholds used are: 0., 0.5, 1.0, 2.0, 5.0, 10.0 and 30.0 mm/h. Pixels above the threshold are labeled as 1 and pixels below are labeled as 0. This allows us to compute a confusion matrix (also known as a contingency table; see Table 6.1).

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Table 6.1: Confusion Matrix

By using this matrix we calculate various verification scores that are commonly used in meteorology. The scores that are used are the Probability of Detection (POD; also called recall), the Critical Success Index (CSI), the False Alarm Ratio (FAR) and the bias (for more information see [65, Chapter 8.2]). These metrics are defined as follows:

$$POD = \frac{TP}{TP + FN} \tag{6.3}$$

$$FAR = \frac{FP}{TP + FP} \tag{6.4}$$

$$CSI = \frac{TP}{TP + FN + FP} \tag{6.5}$$

$$bias = \frac{TP + FP}{TP + FN} \tag{6.6}$$

The CSI (also called threat score) and the bias can be expressed in terms of POD and the success ratio (SR = 1 - FAR).

$$CSI = \frac{1}{\frac{1}{SR} + \frac{1}{POD} - 1} \tag{6.7}$$

$$bias = \frac{POD}{SR} \tag{6.8}$$

In a performance diagram, this relation is used to visualize all 4 terms in a single plot [49]. Figure 6.1 shows an example of such diagram. The x-axis is the SR (1-FAR) and the y-axis indicates the POD score. Lines are drawn to indicate different biases (dashed lines) and CSI values (curved lines). The optimal model would lay in the upper right corner.

6.2.1 Fractions Skill score

The Fractions Skill Score (FSS) is a spatial verification measure that assesses the performance of forecasts on different scales for a given rain rate threshold [48]. In order to compute the FSS a window with a scale of n is moved across the image to compute the fraction of pixels inside the window with a value of 1 (i.e., if the pixel has exceeded the threshold). This results in a matrix of fractions in the observed image $O(n)$ and in the forecast image $F(n)$. The FSS is then defined as:

$$FSS(n) = 1 - \frac{MSE(n)}{MSE(n)_{ref}} \tag{6.9}$$

where $MSE(n)$ is the MSE between $O(n)$ and $F(n)$ and the reference, $MSE(n)_{ref}$, is the largest possible MSE between the observation and the forecast:

$$MSE(n)_{ref} = \frac{1}{N_x N_y} \left[\sum_{i=1}^{N_x} \sum_{j=1}^{N_y} O(n)_{i,j}^2 + \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} F(n)_{i,j}^2 \right] \tag{6.10}$$

where N_x and N_y are the number of rows and columns in the radar data and i and j indicate the row and column index of the fraction matrices, respectively.

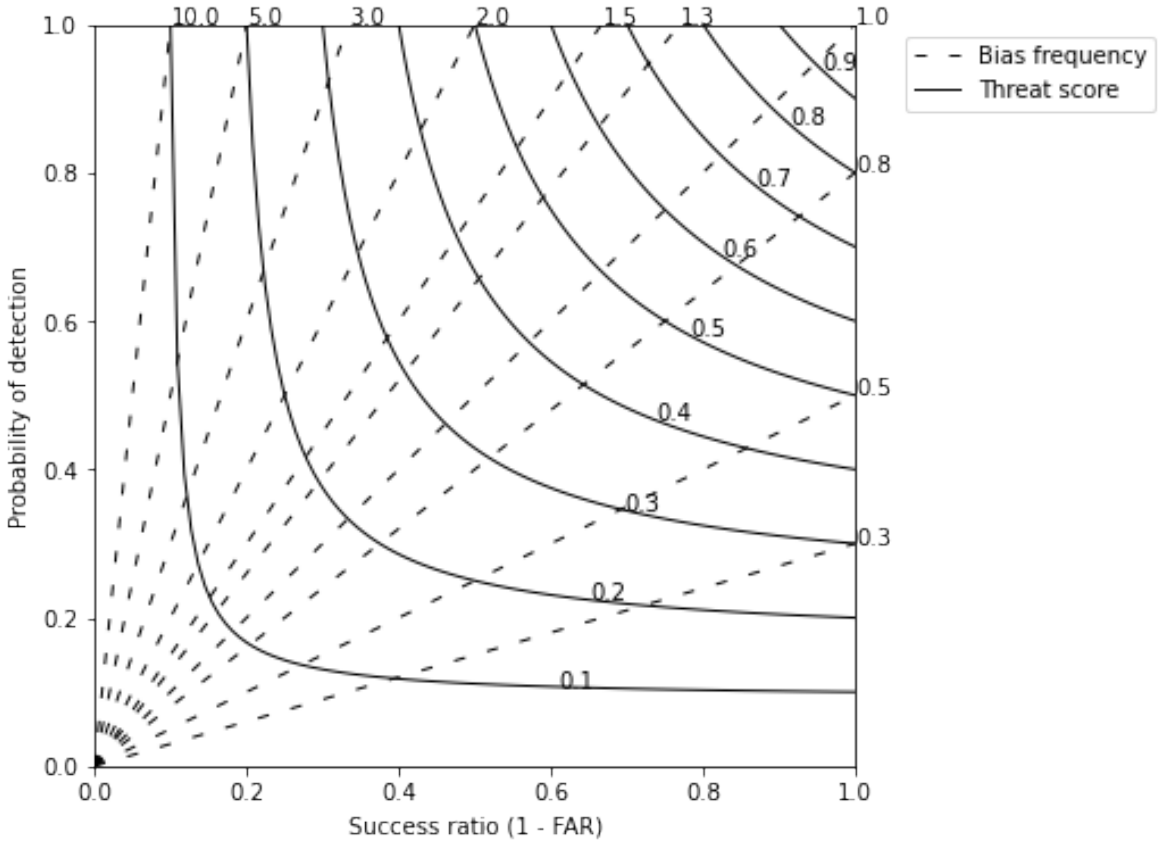


Figure 6.1: This figure shows a performance diagram. The SR and POD are plotted on the x- and y-axis, respectively. The dashed line shows the bias frequencies and the solid lines shows the CSI scores.

Dataset	Period	Nr. Sequences
Training	2008-01-01 till 2018-12-31	67105
Validation	2019-01-01 till 2019-12-31	6570
Testing	2020-01-01 till 2020-12-31	6324

Table 6.2: Dataset for training, validation and testing

Dataset	Period	Nr. Sequences
Training	2008-01-01 till 2020-12-31	67105
Testing	2008-01-01 till 2020-12-31	6324

Table 6.3: Randomly split dataset for training and testing. The training and testing samples were randomly selected without replacement.

The FSS can range from 0 to 1, with 1 being a perfect forecast. As the scale increases the FSS generally increases as well. A model is seen as skillful at lead time t if its FSS score at t is higher than the skill of a random forecast, $FSS_{random} = 0.5 + \frac{f_0}{2}$, where f_0 is the window size divided by the size of the domain. The FSS score allows us to determine for each model the maximum skillful lead time at different scales for different rain intensity thresholds.

6.3 Train Test Split

The authors of the AENN model randomly split their data into a training, validation and test set. However, as we are dealing with time series data, splitting the data at random can be problematic. When the model is trained using data that is not available in a real scenario, data leakage occurs. Realistically, the model cannot have seen future data, therefore its validation when data leakage is present will overestimate its performance in a real-world scenario. For the purpose of preventing data leakage, we separated the dataset chronologically. The first 11 years of the dataset were used for training. The most recent available year (2020) was used as a testing set and the year before that as a validation set. The number of rain events per set can be seen in Table 6.2

Furthermore, we compare the performance when using random splitting instead of chronological splitting to emphasize the importance of splitting time series data correctly. The expectation is that randomly splitting will result in data leakage, which will result in the model obtaining a better score on its test set than a model that was trained and tested on chronologically split data. We randomly selected the same number of training and testing samples for the randomly split dataset (Table 6.3) as we did for the chronologically split dataset.

Chapter 7

Results

In this chapter we present the experimental results. In particular, we compare the forecasting performance of the proposed GAN model versus the baseline model S-PROG and the non-adversarial neural network (NAG). Additionally, we investigate how splitting the data for cross-validation affects the forecasting performance of the GAN model. Furthermore, we show some examples of nowcasts made by the different models to illustrate their differences. All results shown in this chapter are for nowcasts made on the independent test set. No more changes were made to the model from this point on.

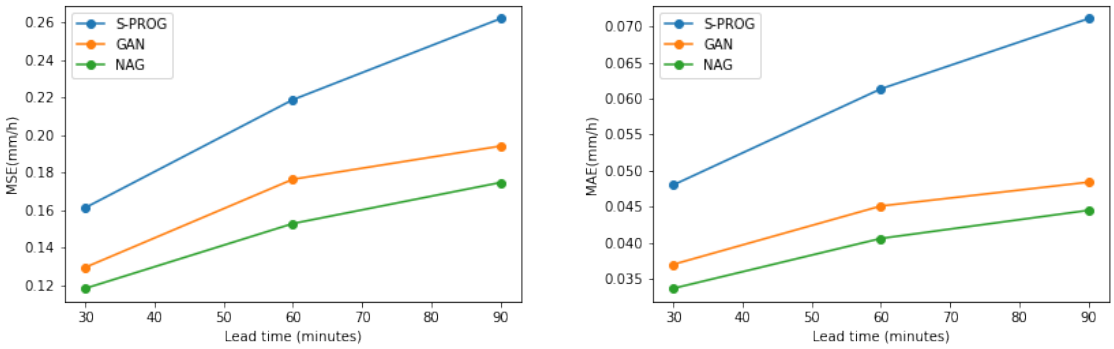
7.1 Nowcasting Performance

In this section we compare the performance of the GAN with the baseline models S-PROG and NAG in terms of forecasting performance. Performance was measured with continuous scores, MSE and MAE, as well as with the categorical scores CSI, FAR, POD and bias. Furthermore, we show the MSE and MAE by location on the radar map to visualize where the models makes errors. Lastly, we computed the fractions skill score to obtain the maximum skillful lead times for forecasting different rain intensities at various scales.

Figure 7.1 shows the performance of all models in terms of their MSE and MAE for different lead times. The S-PROG model obtained the worst MSE and MAE scores on all thresholds. The NAG model performed the best out of all three models at all lead times, followed by the GAN model.

Furthermore, we visualize the MSE and MAE per location for the different models. This can be seen in Figure 7.2a. In order to better show the complete error landscape the colorbar was set to range from 0 to the 99.9 percentile of the MSE or MAE across all models. The outliers are seen as bright yellow dots in this figure. This shows that there is still some clutter in the data and that all the models have trouble dealing with this. Furthermore, all models show a greater average error near the radars in Herwijnen and Den Helder.

The differences in MSE and MAE per location is shown in Figure 7.2b. Here the colorbar shows the range from the 0.1 percentile till the 99.9 percentile of the MSE or MAE of the models. The MAE pattern of the GAN shows that the model tends to wrongly predict precipitation in the lower right corner of the image, which is also visible in Figure 7.2a. Furthermore, a grid pattern of dots is visible where the model performance worse than in the rest of the image. Furthermore, an area in the top left corner is visible where the ML models



(a) (b)

Figure 7.1: Performance of the models (S-PROG, GAN, NAG) measured with continuous scores in mm/h (a) mean squared error, (b) mean absolute error

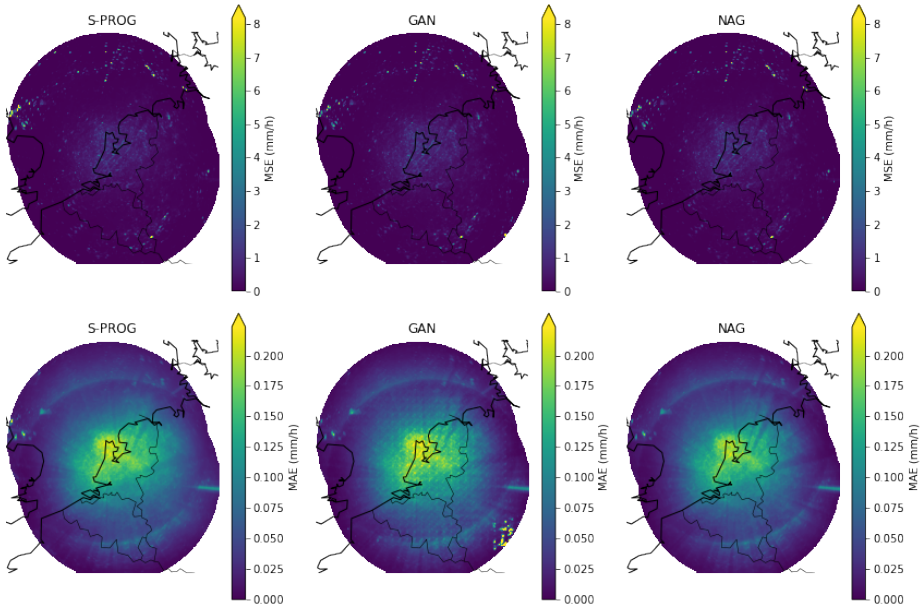
CSI Threshold (mm/h)	0.0	0.5	1.0	2.0	5.0	10.0	30.0
S-PROG	0.487	0.350	0.280	**0.189**	**0.073**	**0.022**	**0.003**
GAN	**0.494**	0.354	**0.283**	0.182	0.054	0.011	0.000
NAG	0.302	**0.363**	0.263	0.137	0.022	0.001	0.000

Table 7.1: CSI scores of the different models on different precipitation rate thresholds. The best results are indicated in bold, the second best results are underlined.

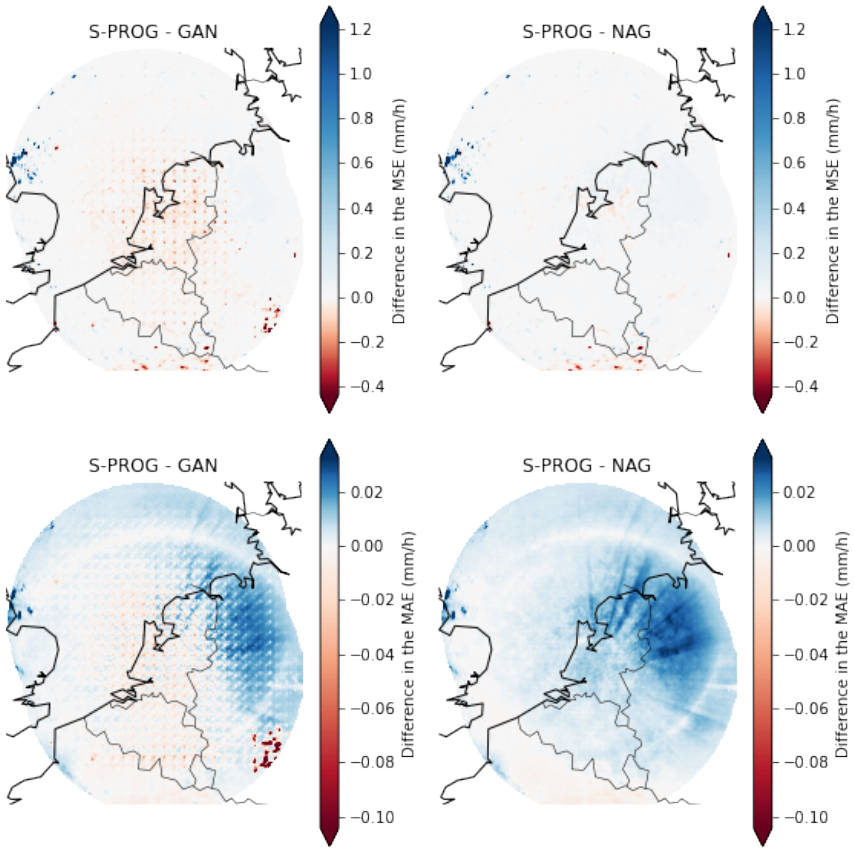
outperform S-PROG.

Furthermore, Table 7.1 shows the CSI score of the models for different thresholds of precipitation. A higher CSI score means a better performance. The GAN model performed best on predicting rain versus no rain (threshold of 0.0 mm/h). For a threshold of 0.5 mm/h the NAG model obtained the best result and S-PROG performed the worst. However, the S-PROG model performed the best on predicting higher rain intensities equal to or above 2.0 mm/h. The GAN model outperforms the NAG model on all precipitation thresholds except 0.5 mm/h.

Additionally, in Figure 7.3 performance diagrams show the POD, SR, CSI and bias of the models at different thresholds and lead times. The rain rate thresholds of 10.0 mm/h and 30.0 mm/h were excluded as none of the models are skillful at predicting those and the POD and SR are close to 0 for those thresholds. The further to the right the higher the success ratio of the model and the further to the top the higher the POD. Deviations from the diagonal indicate the bias of the model. A model has a positive bias when it is above the diagonal and a negative bias if it is below the diagonal (see Chapter 6 for more information). In this diagram the model closest to the top right corner can be seen as the best model. The diagrams show that rainfall intensities of 0.5 mm/h or higher are underestimated by the models and this bias becomes larger as the rain threshold increases.



(a)



(b)

Figure 7.2: (a) MSE (upper row) and MAE (lower row) per location, (b) Difference in MSE (upper row) and MAE (lower row) between GAN and S-PROG (left column) and NAG and S-PROG (right column)

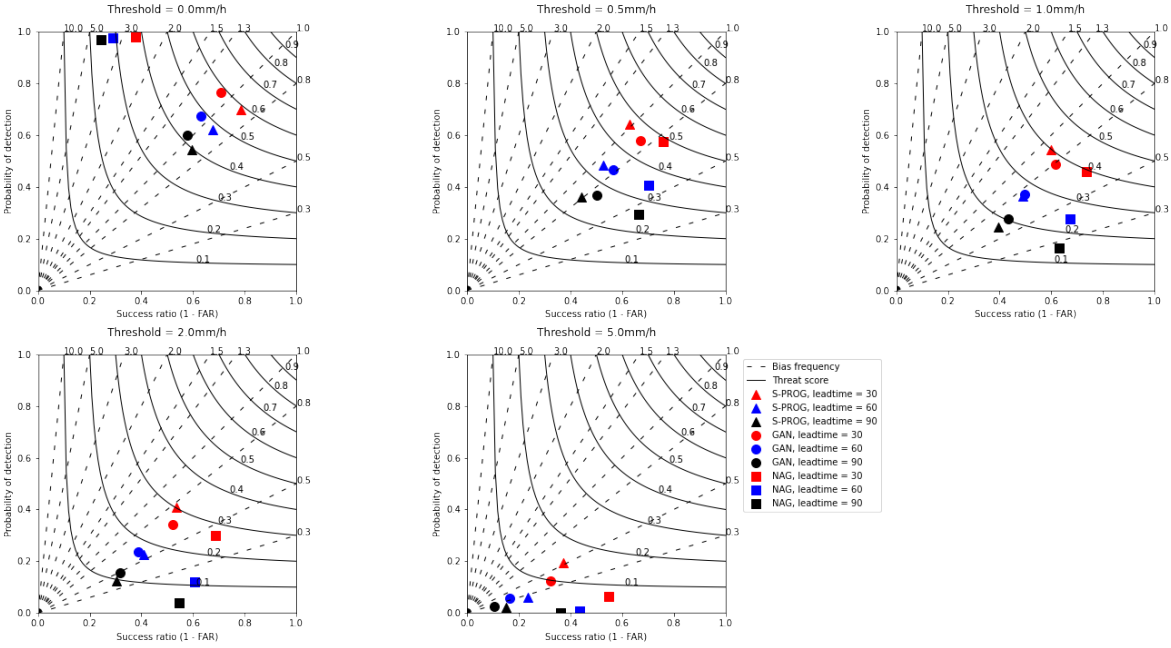
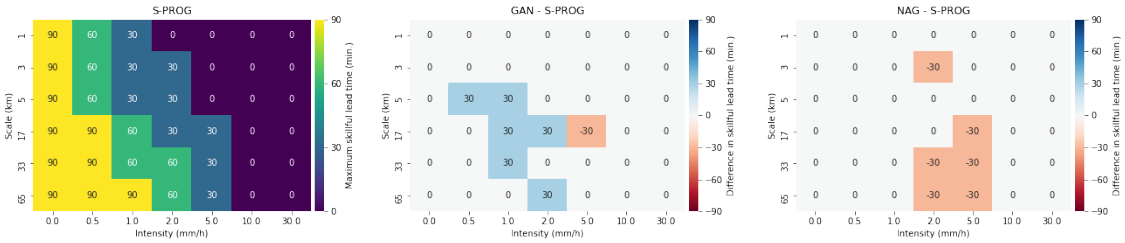


Figure 7.3: Performance diagrams for different thresholds. S-PROG is depicted by triangles, the GAN model by circles and the NAG model by squares. The colors indicate different lead times (red = 30min., blue = 60min., black = 90min.)

Lastly, the FSS was computed for the scales of 1, 3, 5, 17, 33 and 65 km, for the three different lead times and on rain intensity thresholds of 0.0, 0.5, 1.0, 2.0, 5.0, 10.0 and 30.0 mm/h. The maximum skillful lead time of a model for each scale and intensity is the highest lead time that reaches a skillful forecast ($FSS > 0.5 + \frac{f_0}{2}$; subsection 6.2.1). Figure 7.4a shows the maximum skillful lead times of the baseline S-PROG model. Figure 7.4b and 7.4c respectively show the difference in maximum lead times between S-PROG and GAN and S-PROG and NAG. None of the models are skillful at predicting 10 or 30 mm/h at the considered lead times of 30 min. and higher. Rain intensities of 2.0 mm/h and 5.0 mm/h are only possible to be skillfully predicted at a lead time of $\geq$ 30 min. on a scale of $\geq$ 2 km and $\geq$ 16 km, respectively. The GAN model increases the maximum skillful lead times, compared to S-PROG, of predicting 0.5 mm/h on a scale of 5 km, 1.0 mm/h on the scale of 5, 17 and 33 km and also increase the skillful lead time on 2.0 mm/h on scales of 17 and 65 km by 30 minutes. However, its maximum lead time decreases by 30 minutes, relatively to S-PROG, in predicting rain thresholds of 5.0 mm/h on a scale of 16 km. The skillful lead time of the NAG model is 30 minutes lower than S-PROG when predicting rain rates of 2.0 mm/h on scales of 3, 33 and 65 km, and when forecasting rain rates of 5.0 mm/h on scales of 17, 33 and 65 km.

7.2 Influence of Data Splitting

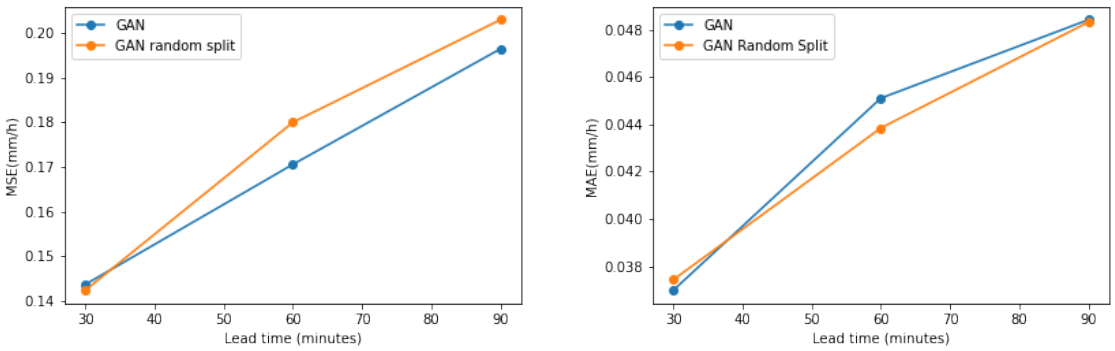
In this section we discuss the influence of splitting the data into training, validation and testing sets at random or chronologically. In the original AENN paper [33] the model was trained and tested on a random split of training and testing data. However, this leads to



(a) (b) (c)

Figure 7.4: A model is seen as skillful at lead time t if the FSS score is higher than $0.5 + \frac{f_0}{2}$ (subsection 6.2.1)
(a) Maximum skillful lead time S-PROG, (b) Difference in skillful lead times between GAN and S-PROG, (c) Difference in skillful lead times between NAG and S-PROG

an unrealistic situation. The model's training set contains information about future samples relative to the samples in the test set. In a real-world scenario the model would not have any information about future samples. This phenomenon is called data leakage (see Chapter 6.3). In order to make a realistic training situation, we split the dataset across time. In this section we compare the results for the model trained on randomly split data versus the model trained on chronologically split data. Figure 7.5 compares the performance of a model trained on a randomly split training-testing data (GAN Random Split) versus the GAN model that was trained on chronologically split data (GAN). No significant difference in performance between the random splitting and the chronological splitting method. The two models obtain a similar performance in terms of MAE on a lead time of 30 and 90 minutes and in terms of MSE on lead time of 30 minutes. Furthermore, the GAN Random Split performs worse in terms of MSE on lead times of 60 and 90 minutes, and the GAN performs worse in terms of MAE on a lead time of 60 minutes.



(a) (b)

Figure 7.5: Performance of the GAN trained and tested on the chronologically split dataset(GAN) versus when trained on randomly split training and testing set(GAN Random Split), (a) mean squared error, (b) mean absolute error

7.3 Visualization of the Nowcast Methods

In this section we give some examples of nowcasts made by the different models. The S-PROG and NAG model both show signs of blurring especially for later lead times of 60 and 90 minutes. The GAN model does not seem to produce blurry predictions. However, at lead times of 60 and 90 minutes the GAN starts to wrongly forecast precipitation in the lower right corner of the image. The visualization of the MAE per location in Figure 7.2 also showed that the GAN tends to make mistakes in this area.

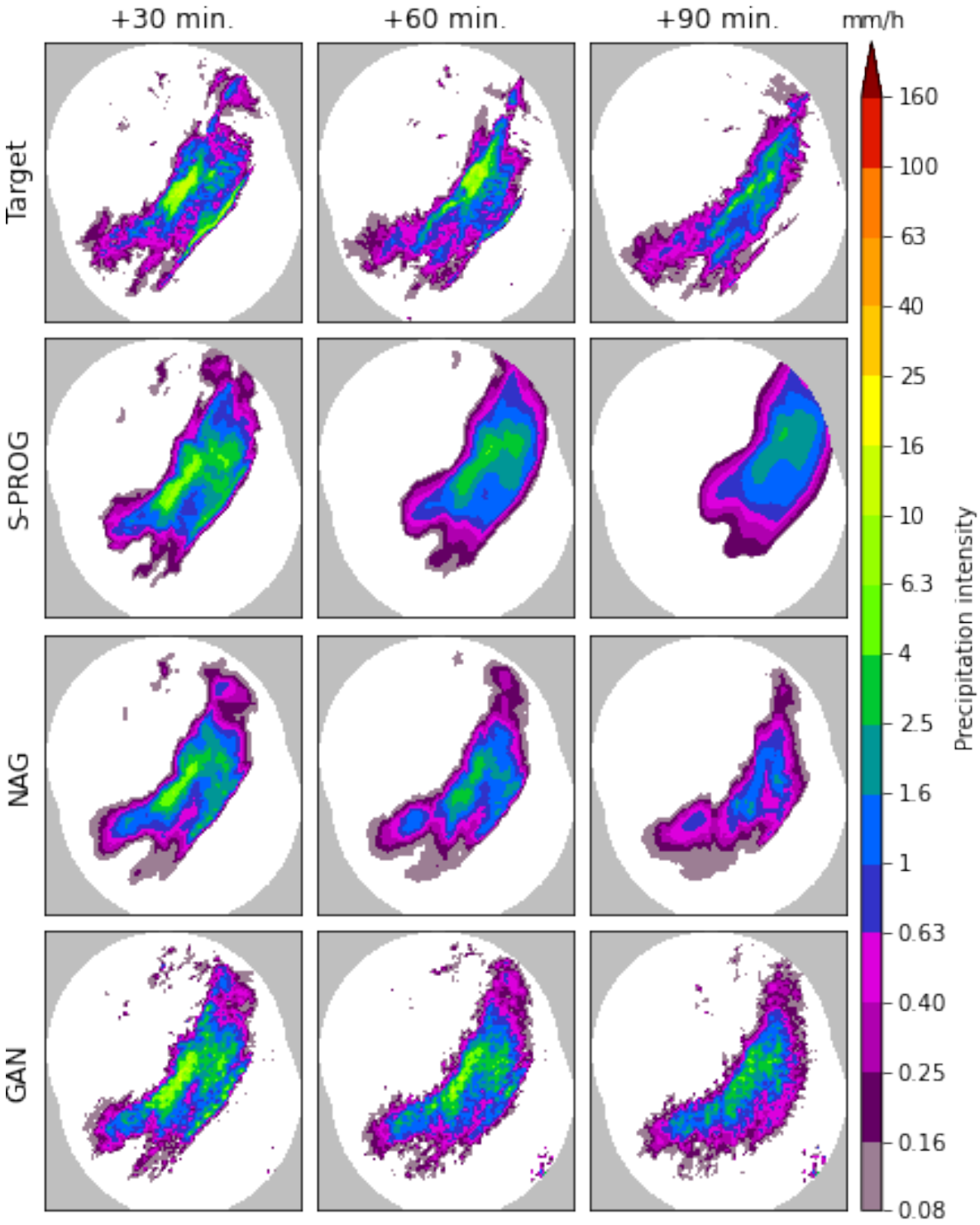


Figure 7.6: Example of nowcasts made by the 3 different models for a rainy event that initiates at 2020-02-16 19:00-19:30. The target images and the predictions are shown for the lead times 30, 60 and 90 minutes (20:00, 20:30 and 21:00, respectively)

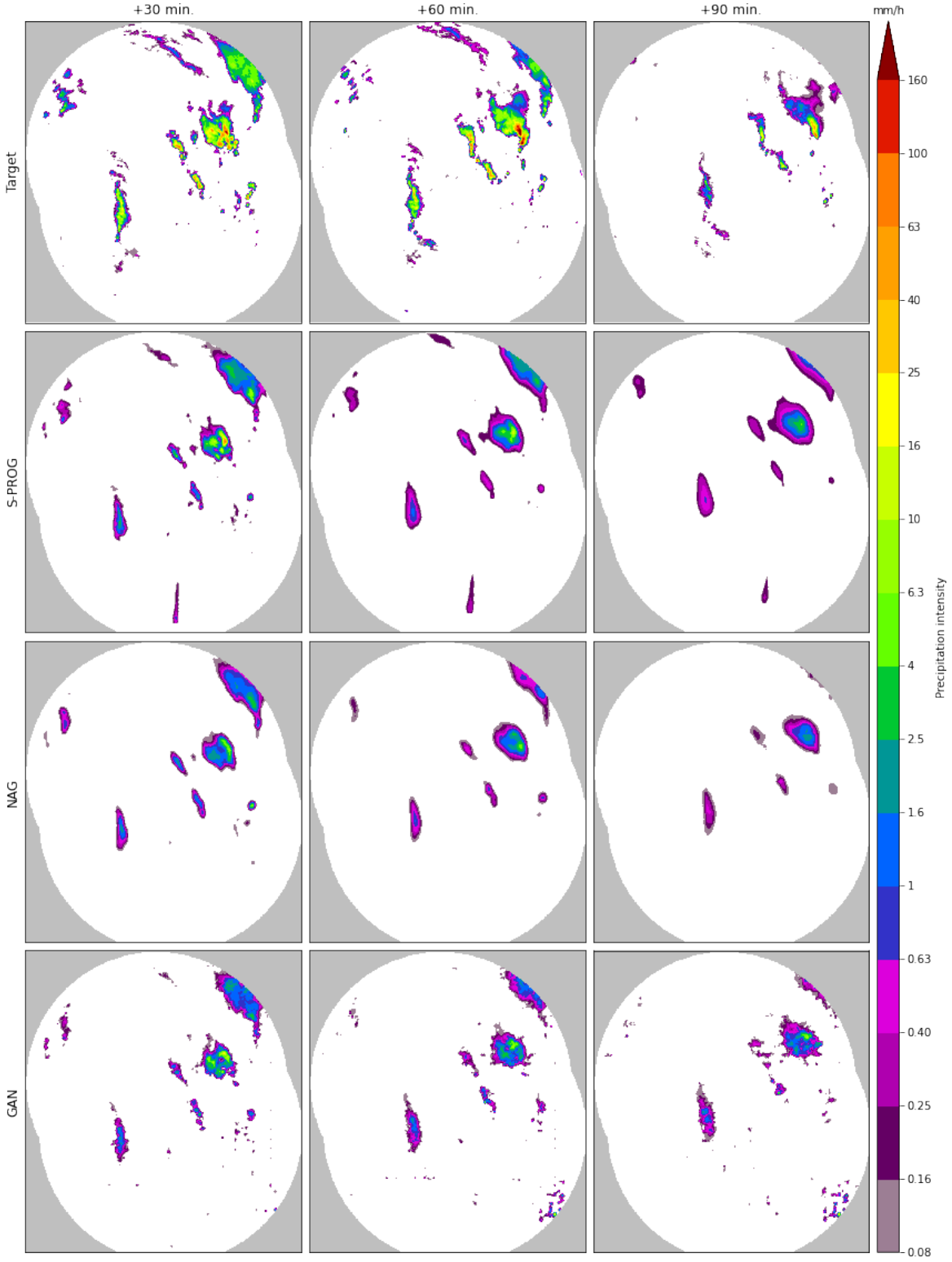


Figure 7.7: Example of nowcasts made by the 3 different models for a rainy event that initiates at 2020-06-27 15:00-15:30. The target images and the predictions are shown for the lead times 30, 60 and 90 minutes (16:00, 16:30 and 17:00, respectively)

Chapter 8

Discussion

Our results showed that the machine learning GAN model scores higher on predicting light rain ($\leq$2.0 mm/h) in terms of the CSI score for the lead times of 60 and 90 minutes. This is consistent with the work of Jing et al. [33] who also showed that the AENN GAN model outperformed traditional extrapolation methods when evaluating on a threshold of 0.5 mm/h. However, a more extensive evaluation showed that the baseline optical flow model S-PROG performs better than the ML methods on predicting heavier precipitation of more than 2.0 mm/h, while Jing et al. only tested the nowcasting performance at 0.5 mm/h. Given that heavier rainfall has a bigger impact and is more dangerous than lighter rain, it is important to take into consideration metrics that measure the model's performances at higher rain rate thresholds. Though the AENN model does not perform well on predicting rain intensities above 2.0 mm/h, we also showed that even a state-of-the-art radar extrapolation method cannot skillfully predict such high rain intensities at a scale of 1 km at lead times of 30 minutes or more. This goes to show how difficult this task is. However, the ConvGRU GAN model (GA-ConvGRU) proposed in [59] was reported to outperform an optical-flow based baseline method for the higher rain intensities. The GA-ConvGRU has over 20 million trainable parameters while the AENN model has close to 2 million trainable parameters. The increased capacity of the GA-ConvGRU might be an explanation for the better performance at higher precipitation thresholds. Future work needs to be done to investigate if increasing the model's capacity improves forecasts of heavier precipitation.

Furthermore, Figure 7.1 showed that the two machine learning models, GAN and NAG, both outperformed the optical flow model S-PROG in terms of MSE and MAE. However, the low MSE and MAE of the ML models are likely explained by their ability to predict light rain ($\leq$2.0 mm/h), which is much more common than the heavier precipitation intensities. On the basis of MSE and MAE, the S-PROG performance is the worst. However, the CSI scores reveal that the S-PROG algorithm is better at predicting moderate to high intensity precipitation. The continuous scores do not provide a complete picture of the performance of a precipitation nowcasting model, so it is vital to evaluate such model with multiple metrics. Furthermore, these findings also show the downsides of using MSE as a loss function for precipitation nowcasting. Optimizing for MSE leads to the model only learning to predict light precipitation well, while in a real-world scenario the ability to forecast heavy rain intensities is important. We showed that the adversarial loss improves the forecasts of higher rain intensities which is consistent with the work in [26]. Another study proposed the balanced

MSE (B-MSE) and balanced MAE (B-MAE) as a loss function for precipitation nowcasting [54]. In the B-MSE and B-MAE more weight is assigned to heavier precipitation during the calculation of the MSE and MAE. The balanced MSE (B-MSE) and balanced MAE (B-MAE) was reported to increase the model's performance on predicting heavier rainfall.

Furthermore, the visualization of forecasts of the different models (Chapter 7.3) demonstrated that a model trained on only MSE (i.e. NAG) showed blurry forecasts which are unrealistic. This is also the case for extrapolation-based models like S-PROG. The predictions of the GAN model were not blurry.

In summary, we can conclude that the MSE does not seem to correlate well with human judgement of what makes good forecasts. Our findings are in line with previous research that showed that optimizing for a low MSE results in blurry output [42] and a bias towards predicting low rain intensity thresholds [54]. The adversarial loss was shown to help with both of these problems. Further research on precipitation nowcasting should try to avoid optimizing MSE and instead use a loss that encourages the model to learn to predict higher rain intensities better, like for example the B-MSE [54].

Visualization of the average error per pixel revealed that there are some strange artifacts present in the output of the GAN model (see Figure 7.2). Firstly, Figure 7.2b showed that the errors of the GAN form a repeated pattern like a grid. We do not know for sure what caused this. We speculate that these artifacts are caused by the transposed convolution layers. Studies have reported that these can cause a checkerboard like pattern to emerge in the neural network's output [45]. Applying bilinear interpolation upscaling to this pattern might transform the checkerboard artifacts into the form of a grid. The generator with no adversarial loss did not show this pattern. However, it is possible for a neural network to learn to adjust its weight in order to avoid the checkerboard artifact [45]. We suspect the GAN model was not able to learn to do this. Resize-convolution layers have been shown to eliminate checkerboard artifacts [45]. Further research needs to be done to see if the grid like error pattern of the GAN disappears if resize-convolution layers are used instead of transposed convolution layers.

Secondly, the MAE pattern of the GAN model (Figure 7.2a) showed an area in the bottom right corner where the GAN tends to make mistakes. A closer inspection of the GAN nowcasts (Figure 7.6 and Figure 7.7) shows that the GAN predicted the same precipitation field in this area for lead times of 60 and 90 minutes for the two different rain events. The generator appears to have learned to always predict the same precipitation amount in the same part of the image in order to fool the discriminator. This artifact is not present during all stages of the training of the GAN model (see Appendix A.5). One way to avoid this is to visually inspect the output of the GAN model during training and select the model with the best reconstruction loss that does not show artifacts.

The error patterns showed that there is still some clutter in the dataset. In Figure 7.2a outliers in terms of MSE per pixel can be seen in areas over sea; these are likely pixels that have high probability of containing clutter. All models show a high MAE at an area near the coast of England. Around this area offshore wind turbines are located [14]. We speculate that the errors in the models occur because of the clutter caused by the wind turbines. The ML methods have a lower average error around this area (see Figure 7.2b). The ML models have an unfair advantage in this respect, as they can learn to predict clutter and extrapolation-based methods cannot do this.

Furthermore, the error maps for the models show certain patterns and structures. These patterns are most apparent in the Figure 7.2a, which shows the MAE per location. On average the models make the largest mistakes at areas close to the radars in Herwijnen and Den Helder. Bright band may be responsible for this. Bright band is a region of enhanced reflectivity associated with the melting of snow [4, 25]. The melting of snow happens at a specific elevation and when the radar beam moves through this layer it will pick up an increased reflectivity. As the radar scans in an upward direction, this can produce a circular band of higher reflectivity around the radar. The radar overestimates reflections near itself, so the errors there are also greater. Furthermore, one can see a circular curve further away from the radar in the North Sea and across Belgium and Germany. We speculate that these artifacts can be explained by the fact that the radars have limited range. The circular curves are likely associated with the transition from two radars to one radar.

Furthermore, originating from the radars location cone-shaped areas can be seen where the models' errors are lower. These are probably areas where the reflectivity is underestimated due to blockage of the radar beams.

Ideally, the real-time radar product would have been further corrected for bias and cluttered. Reducing the clutter in the dataset would make the comparison between ML and extrapolation-based methods fairer. One way to do this would be to validate over the area above the Netherlands. Most clutter is above sea so this would reduce the amount of clutter in the validation set. Furthermore, a more extensive bias-correction has been done on one of KNMI's radar products by using manual gauges. This dataset is however not available in real time. Future research should focus on the image-to-image translation from the real-time radar data to the further bias-corrected radar data. In this thesis we initially tried to do this, but the machine learning models were not able to perform this task well (Appendix A.4). We speculate that extra information like temperature and wind speed is needed to do well on this task. Given the time constraints of this research, we instead focused on forecasting the real-time radar data.

Furthermore, the Fraction Skill Score (FSS) was used to estimate the maximum skillful lead times for the model for different rain intensities and scales. This revealed that none of the models are skillful at predicting rain of intensities above 2.0 mm/h at a scale of 1 km. While S-PROG outperforms the other models on moderate to high precipitation, a random forecast is still better at forecasting rain above 2.0 mm/h on lead times of 30 or more minutes. However,our results are limited to the lead times that the AENN model was designed for, which are 30, 60, and 90 minutes. Recent research showed that S-PROG is able to skillfully predict rain intensities above 2.0 mm/h on lead times shorter than 30 minutes [61]. Further research should include shorter lead times with smaller intervals between them. This would give a more detailed view of the differences between the models.

It was expected that randomly splitting the dataset into training and testing would lead to leakage and thus result in a better performance on the test set. However, against expectations the model trained and tested on randomly split data did not perform significantly better than the model trained on chronologically split data. We speculate that the reason that the random split model does not perform better is that the amount of leakage is limited. The rainy events were labeled (see Chapter 3) such that only 38% percent of the 30 minute sequences are included into the final dataset. Therefore, there tends to be a gap between consecutive samples in terms of time. Precipitation is temporally correlated, so samples closer in time

tend to be more similar to each other than samples far away in time. Leakage would occur when rainy events close in time end up in training and testing datasets and it is likely the case that such cases are limited. Thus the amount of data leakage would also be limited, which would explain why the model trained on randomly split data did not obtain a significantly better validation score when compared to the other model.

Chapter 9

Conclusion

In this thesis we investigated the use of generative adversarial networks to nowcast precipitation in the Netherlands. A GAN model based on AENN [33] was implemented and trained on historical Dutch weather radar data. The GAN's loss function consists of a combination of adversarial loss and reconstruction loss (MSE). Furthermore, we presented a method to reduce the amount of clutter in the radar dataset. The GAN model was compared to the optical flow algorithm S-PROG [52]. Additionally, we analysed the influence of the adversarial loss by comparing two models with the same architecture, where one receives an adversarial loss (GAN) and the other does not (NAG). The machine learning models were trained and evaluated on lead times of 30, 60 and 90 minutes.

The results showed that the GAN model performed the best on forecasting light precipitation of 1.0 mm/h or lower and for moderate to heavy rainfall with intensities above 2.0 mm/h the optical flow model S-PROG performed better. However, neither S-PROG nor the machine learning models were skillful at predicting moderate to heavy rainfall at a scale of 1 km at a lead time of $\geq$ 30 minutes. Furthermore, only the GAN model was able to produce realistic-looking forecasts, the other two models produced blurry forecasts.

Future work on precipitation nowcasting should avoid the use of mean-square-error as a loss function as this causes two problems: 1) it leads to blurry images and 2) it results in a low skill at forecasting moderate to high intensity rainfall. Adding adversarial loss to the machine learning model (GAN) was shown to reduce the blurriness of the predictions and increased the model's performance on moderate to high rain intensities. However, the predictions of the GAN model showed unnatural artifacts, which we believe are related to the checkerboard artifacts common in deep neural networks. Further work is needed to address this problem. In addition, the GAN forecasts moderate to high precipitation intensities worse than the S-PROG model. Further research could improve the GAN's performance on higher rain intensities by using reconstruction loss that assigns more weight to heavier precipitation, like B-MSE or B-MAE [54].

Bibliography

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Aoyagi, J. Ground clutter rejection by MTI weather radar. In *18th Conference on Radar Meteorology* (1978), pp. 358–363.

[3] Attema, J., Bakker, A., Beersma, J., Bessembinder, J., Boers, R., Brandsma, T., van den Brink, H., Drijfhout, S., Eskes, H., Haarsma, R., et al. KNMI'14: Climate change scenarios for the 21st century—a Netherlands perspective. *KNMI: De Bilt, The Netherlands* (2014).

[4] Austin, P. M., and Bemis, A. C. A quantitative study of the "bright band" in radar precipitation echoes. *Journal of Atmospheric Sciences 7*, 2 (1950), 145–151.

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

[6] Bebbington, D., Rae, S., Bech, J., Codina, B., and Picanyol, M. Modelling of weather radar echoes from anomalous propagation using a hybrid parabolic equation method and NWP model data. *Natural Hazards and Earth System Sciences 7*, 3 (2007), 391–398.

[7] Bech, J., Codina, B., and Lorente, J. Forecasting weather radar propagation conditions. *Meteorology and Atmospheric Physics 96*, 3 (2007), 229–243.

[8] Beekhuis, H., and Holleman, I. From pulse to product, highlights of the digital-IF upgrade of the Dutch national radar network. In *Proceedings of the 5th European Conference on Radar in Meteorology and Hydrology, Helsinki, Finland* (2008), vol. 30.

[9] Bihlo, A. A generative adversarial network approach to (ensemble) weather prediction. *Neural networks : the official journal of the International Neural Network Society 139* (2021), 1–16.

[10] Bowler, N. E., Pierce, C. E., and Seed, A. Development of a precipitation nowcasting algorithm based upon optical flow techniques. *Journal of Hydrology 288*, 1-2 (2004), 74–91.

[11] Changnon, S. A. Effects of summer precipitation on urban transportation. *Climatic Change 32*, 4 (1996), 481–494.

[12] Cheng, M., and Brown, R. Delineation of precipitation areas by correlation of Meteosat visible and infrared data with radar data. *Monthly weather review 123*, 9 (1995), 2743–2757.

[13] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Doha, Qatar, Oct. 2014), Association for Computational Linguistics, pp. 1724–1734.

[14] Crabtree, C. J., Zappalá, D., and Hogg, S. I. Wind energy: UK experiences and offshore operational challenges. *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy 229*, 7 (2015), 727–746.

[15] Crum, T., Ciardi, E., and Sandifer, J. Wind farms: Coming soon to a WSR-88D near you. *Nexrad Now 18* (2008), 1–7.

[16] Dai, Y. Post-processing cloud cover forecasts using Generative Adversarial Networks . Master's thesis, Swiss Federal Institute of Technology Zurich, Switzerland, 2020.

[17] De Luca, D. *Rainfall Nowcasting Models for Early Warning Systems.* Nova Publisher: Hauppauge, NY, USA, 2013.

[18] Fadnavis, S. Image interpolation techniques in digital image processing: an overview. *International Journal of Engineering Research and Applications 4*, 10 (2014), 70–73.

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

[20] Glorot, X., and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (2010), JMLR Workshop and Conference Proceedings, pp. 249–256.

[21] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *Advances in Neural Information Processing Systems 3* (06 2014), 2672–2680.

[22] Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations* (2014).

[23] Grecu, M., and Krajewski, W. A large-sample investigation of statistical procedures for radar-based short-term quantitative precipitation forecasting. *Journal of hydrology 239*, 1-4 (2000), 69–84.

[24] Grecu, M., and Krajewski, W. F. An efficient methodology for detection of anomalous propagation echoes in radar reflectivity data using neural networks. *Journal of atmospheric and oceanic technology 17*, 2 (2000), 121–129.

[25] Hall, W., Rico-Ramirez, M. A., and Krämer, S. Classification and correction of the bright band using an operational C-band polarimetric radar. *Journal of Hydrology 531* (2015), 248–258.

[26] Hayatbini, N., Kong, B., Hsu, K.-l., Nguyen, P., Sorooshian, S., Stephens, G., Fowlkes, C., Nemani, R., and Ganguly, S. Conditional generative adversarial networks (cGANs) for near real-time precipitation estimation from multispectral GOES-16 satellite Imageries—PERSIANN-cGAN. *Remote Sensing 11*, 19 (2019), 2193.

[27] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[28] Holleman, I., and Beekhuis, H. *Review of the KNMI clutter removal scheme*. KNMI De Bilt, 2005.

[29] Imhoff, R., Brauer, C., Overeem, A., Weerts, A., and Uijlenhoet, R. Spatial and temporal evaluation of radar rainfall nowcasting techniques on 1,533 events. *Water Resources Research 56*, 8 (2020), e2019WR026723.

[30] Ingram, K., Roncoli, M., and Kirshen, P. Opportunities and constraints for farmers of West Africa to use seasonal precipitation forecasts with Burkina Faso as a case study. *Agricultural systems 74*, 3 (2002), 331–349.

[31] Islam, T., Rico-Ramirez, M. A., Han, D., and Srivastava, P. K. Artificial intelligence techniques for clutter identification with polarimetric radar signatures. *Atmospheric Research 109* (2012), 95–113.

[32] Jensen, T. B., Gill, R. S., Overgaard, S., Hansen, L. K., and Nielsen, A. A. Detecting weather radar clutter using satellite-based nowcasting products. In *Proceedings of the Fourth European Conference on Radar in Meteorology and Hydrology (ERAD)* (2006).

[33] Jing, J., Li, Q., Ding, X., Sun, N., Tang, R., and Cai, Y. AENN: a generative adversarial neural network for weather radar echo extrapolation. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 42* (2019), 89–94.

[34] Joss, J., Waldvogel, A., and Collier, C. Precipitation measurement and hydrology. In *Radar in meteorology*. Springer, 1990, pp. 577–606.

[35] Karras, T., Laine, S., and Aila, T. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 4401–4410.

[36] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[37] Kulesa, G. Weather and aviation: How does weather affect the safety and operations of airports and aviation, and how does FAA work to manage weather-related effects? In *The Potential Impacts of Climate Change on Transportation* (2003).

[38] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 4681–4690.

[39] Lucas, B. D., Kanade, T., et al. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)* (1981), Vancouver, British Columbia.

[40] Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of Machine Learning Research* (2013), vol. 30, Citeseer, p. 3.

[41] Marshall, J., and Palmer, W. The distribution of raindrops with size. *Journal of Atmospheric Sciences 5* (07 1948), 165–166.

[42] Mathieu, M., Couprie, C., and LeCun, Y. Deep multi-scale video prediction beyond mean square error. In *International Conference on Learning Representations* (2015), p. pp. 1–14.

[43] Mirza, M., and Osindero, S. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[44] Nair, V., and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning* (Madison, WI, USA, 2010), ICML'10, Omnipress, p. 807–814.

[45] Odena, A., Dumoulin, V., and Olah, C. Deconvolution and checkerboard artifacts. *Distill 1* (10 2016).

[46] Overeem, A., Holleman, I., and Buishand, A. Derivation of a 10-year radar-based climatology of rainfall. *Journal of Applied Meteorology and Climatology 48*, 7 (2009), 1448–1463.

[47] Pulkkinen, S., Nerini, D., Pérez Hortal, A. A., Velasco-Forero, C., Seed, A., Germann, U., and Foresti, L. Pysteps: an open-source python library for probabilistic precipitation nowcasting (v1. 0). *Geoscientific Model Development 12*, 10 (2019), 4185–4219.

[48] Roberts, N. M., and Lean, H. W. Scale-selective verification of rainfall accumulations from high-resolution forecasts of convective events. *Monthly Weather Review 136*, 1 (2008), 78–97.

[49] Roebber, P. J. Visualizing multiple measures of forecast quality. *Weather and Forecasting 24*, 2 (2009), 601–608.

[50] Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (2015), Springer, pp. 234–241.

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

[52] Seed, A. A dynamic and spatial scaling approach to advection forecasting. *Journal of Applied Meteorology 42*, 3 (2003), 381–388.

[53] Senouci, A., Al-Abbasi, M., and Eldin, N. N. Impact of weather conditions on construction labour productivity in Qatar. *Middle East Journal of Management 5*, 1 (2018), 34–49.

[54] Shi, X., Gao, Z., Lausen, L., Wang, H., Yeung, D.-Y., Wong, W.-k., and WOO, W.-c. Deep learning for precipitation nowcasting: A benchmark and a new model. In *Advances in Neural Information Processing Systems* (2017), I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, Curran Associates, Inc.

[55] Sugier, J., du Chatelet, J. P., Roquain, P., and Smith, A. Detection and removal of clutter and anaprop in radar data using a statistical scheme based on echo fluctuation. *Proceedings of ERAD (2002)* (2002), 17–24.

[56] Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (2014), pp. 3104–3112.

[57] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the Inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 2818–2826.

[58] Thirugnanam, H., Ramesh, M. V., and Rangan, V. P. Enhancing the reliability of landslide early warning systems by machine learning. *Landslides 17*, 9 (2020), 2231–2246.

[59] Tian, L., Li, X., Ye, Y., Xie, P., and Li, Y. A generative adversarial gated recurrent unit model for precipitation nowcasting. *IEEE Geoscience and Remote Sensing Letters 17*, 4 (2019), 601–605.

[60] Tran, Q.-K., and Song, S.-k. Computer vision in precipitation nowcasting: Applying image quality assessment metrics for training deep neural networks. *Atmosphere 10*, 5 (2019), 244.

[61] van der Kooij, E. Nowcasting heavy precipitation in the Netherlands: a deep learning approach. Master's thesis, Delft University of Technology, the Netherlands, 2021.

[62] Venugopal, V., Foufoula-Georgiou, E., and Sapozhnikov, V. Evidence of dynamic scaling in space-time rainfall. *Journal of Geophysical Research: Atmospheres 104*, D24 (1999), 31599–31610.

[63] Wessels, H., and Beekhuis, J. Stepwise procedure for suppression of anomalous ground clutter. In *COST-75 Seminar on Advanced Radar Systems, EUR* (1994), vol. 16013, pp. 270–277.

[64] Wessels, H. R., and Beekhuis, J. Automatic suppression of anomalous propagation clutter for noncoherent weather radars. *NASA STI/Recon Technical Report N 94* (1992), 17407.

[65] Wilks, D. S. *Statistical methods in the atmospheric sciences*, vol. 100. Academic press, 2011.

[66] Xingjian, S., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems* (2015), pp. 802–810.

[67] Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., and Metaxas, D. N. StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 5907–5915.

Appendix A

Appendix

A.1 Clutter-removal using Gradients

Samples are seen as rainy if the average precipitation per pixel is above 0.01 mm/h. Non-rainy samples were discarded as well as rainy samples that have to many high gradients. A pixel is seen as abnormal if it has a gradient of 30 mm/h/km or higher. Below two figures are shown that depict 25 random samples of the rainy samples that were discarded because they have to many abnormal pixels and 25 random rainy samples that were not discarded.

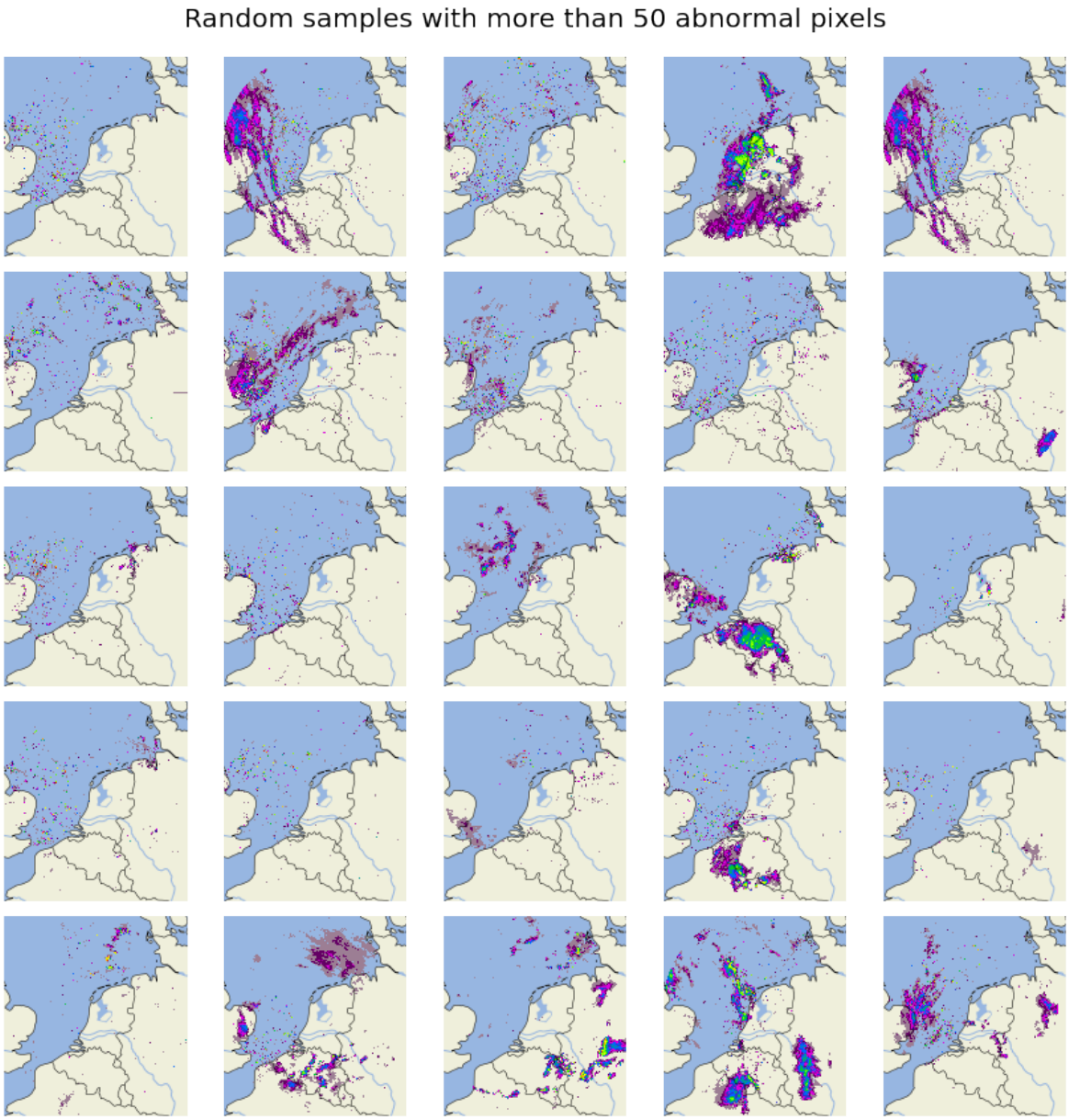


Figure A.1: Random sample of rainy images that were discarded because they contain too many abnormal pixels. Most of the images contain clutter, visible by the small isolated specks (mostly above sea)

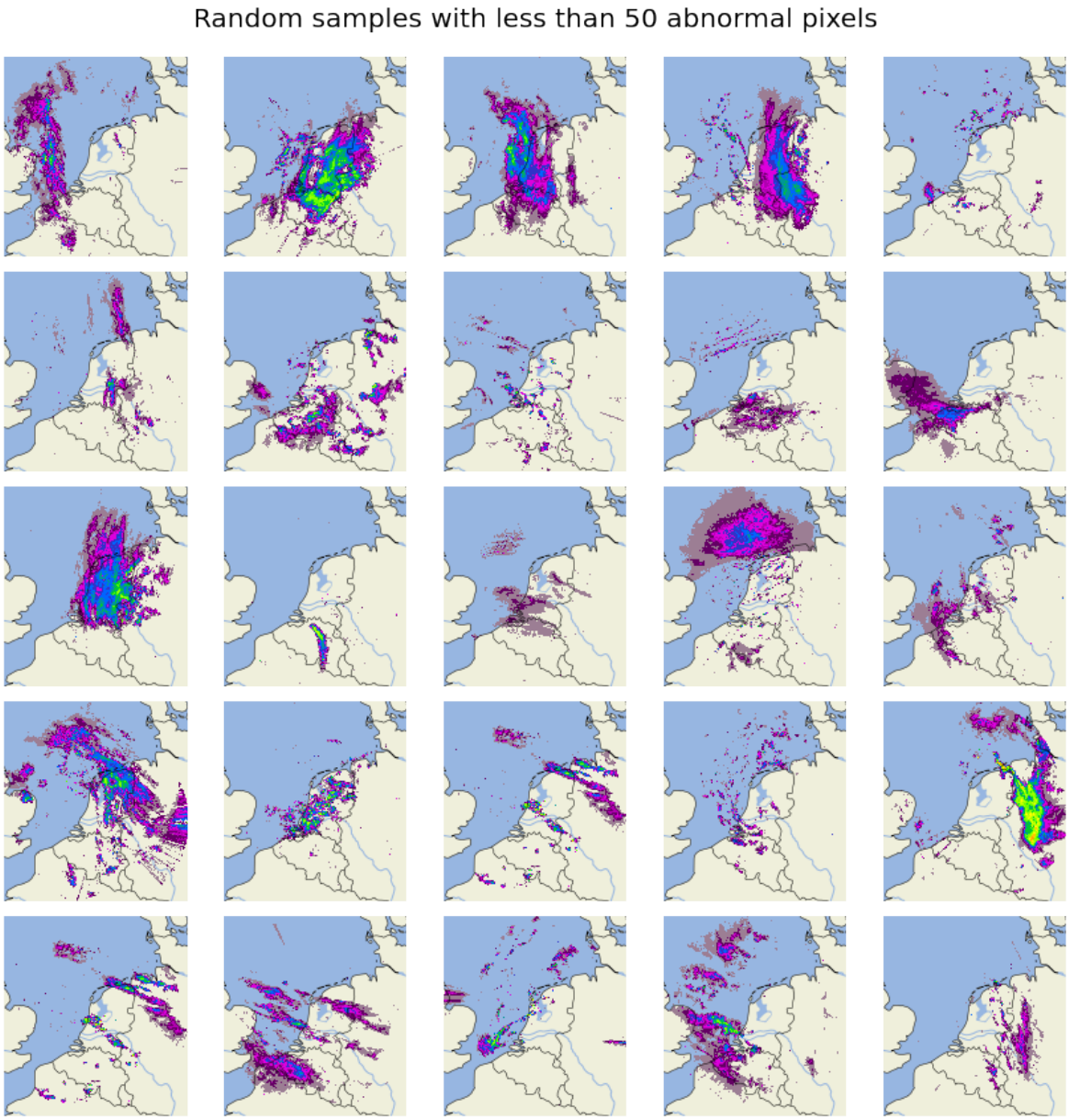


Figure A.2: Random sample of rainy images that do not contain to many abnormal pixels

A.2 Hyperparameter Tuning

A.2.1 Label Smoothing

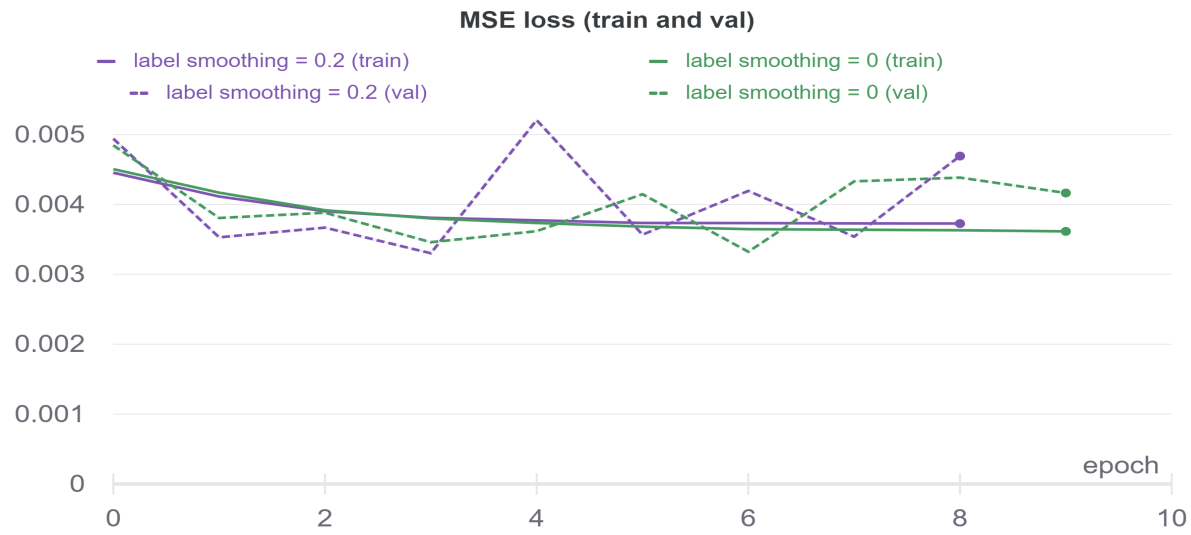


Figure A.3: Performance when using label smoothing (label smoothing factor of 0.2) versus no label smoothing at all (label smoothing factor of 0). Note that the MSE is not terms of mm/h but it is calculated on the normalized dBZ values. The models were run for 10 epochs with early stopping after 5 epochs of no improvement.

A.2.2 Learning Rate

Figure A.4: Comparison of the model performance when using different learning rates (lr). The MSE is calculated on the normalized dBZ values

A.3 Artifact in GAN output

The GAN seems to have learned to forecast the same precipitation in the bottom right corner of the image in order to fool the GAN. By looking at the predictions at two different points during training we can see that this behavior is not present during all stages of the training (see Figure A.5). The best reconstruction loss was obtained by the GAN after 5 epochs of training, however this model shows artifacts in bottom right corner which were revealed by plotting the average MAE per pixel (Figure 7.2a).

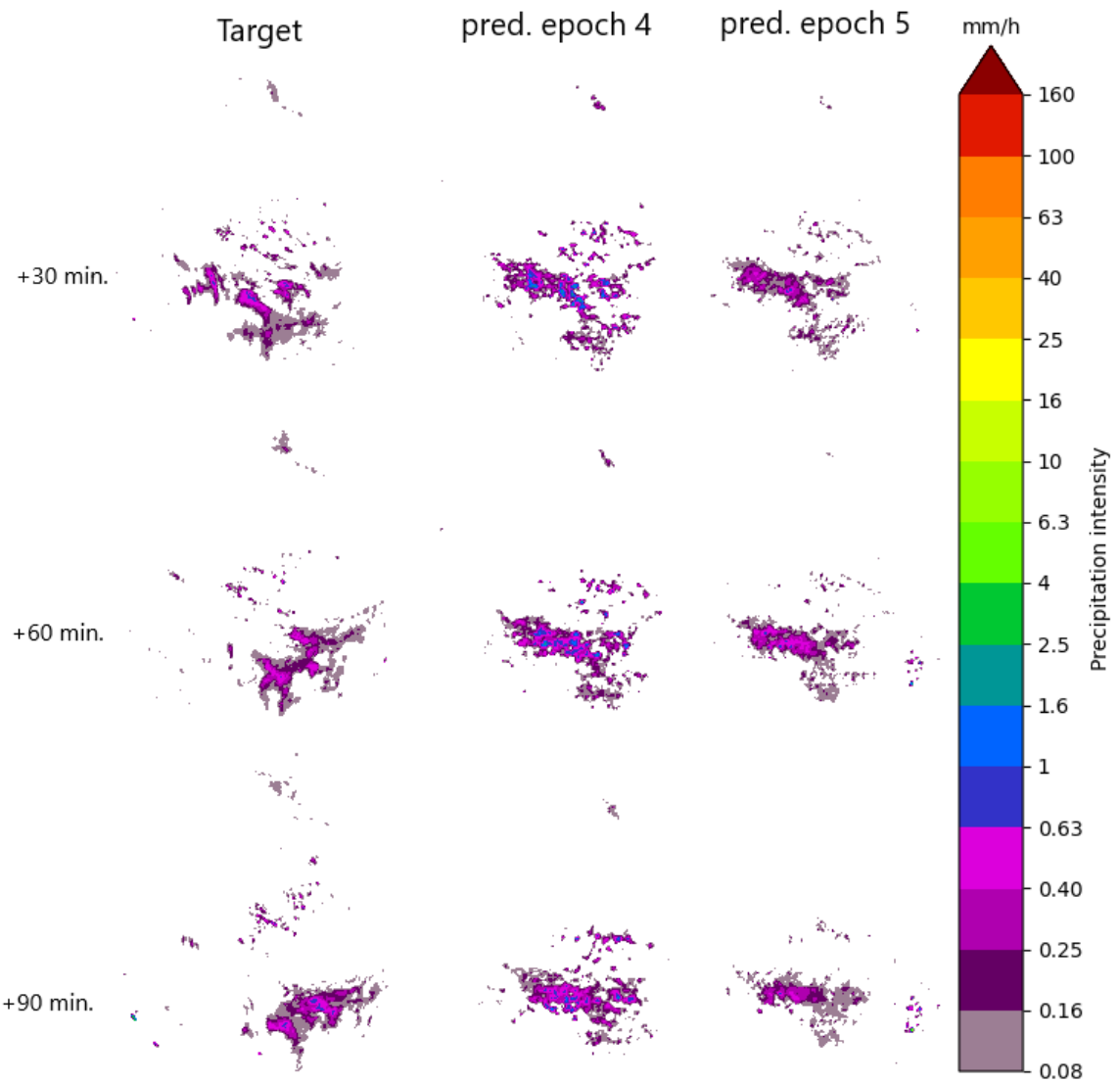


Figure A.5: Visualization of a target in the validation set (left) and the predictions of the GAN model after 4 epochs (middle) and 5 epochs (right) of training

A.4 MFBS dataset

During this research project, we also tried to forecast another radar product given the real-time radar observations. The other radar product is not available in real time. This dataset, called MFBS, contains data that was corrected using automatic and manual rain gauges [46][1]. This dataset gives a more accurate measurement of precipitation in the Netherlands than the RT dataset. This dataset is not available in real time as the manual rain gauge values are only updated once per day. Furthermore, the dataset is only available over a limited area, namely the Netherlands plus an additional few tens of kilometers over the borders and over water. Figure A.6 shows a side by side comparison of a radar scan from the two datasets on the same time.

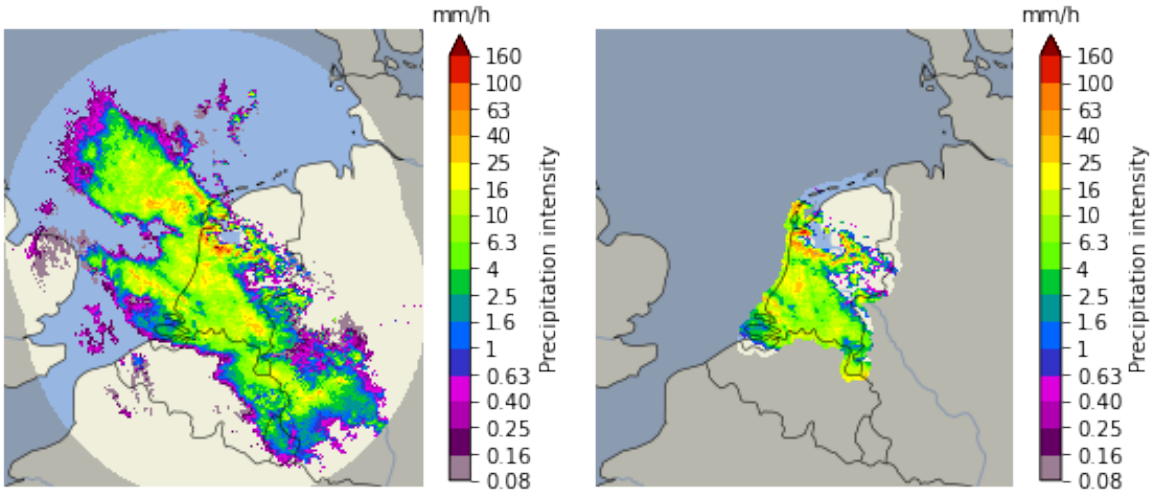
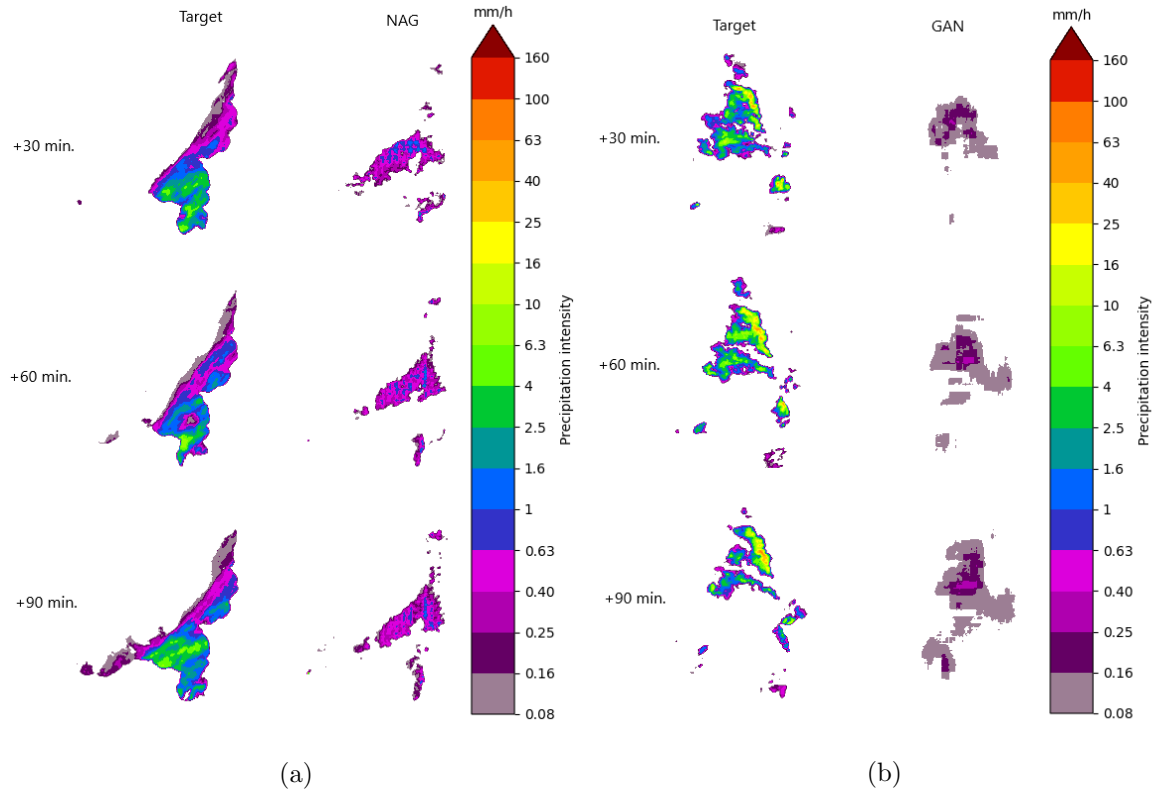


Figure A.6: Example of real time radar scan (left) and MFBS radar data (right), date-time: 2019-06-06 23:50.

The MFBS dataset was preprocessed just like the RT dataset however an extra preprocessing step was performed. The MFBS dataset has the same size as the RT dataset, however the MFBS dataset is masked such that only pixels over the Netherlands can have a non-zero value. As a result, many pixels in the image are redundant since they are always masked. We took a center crop of 384 by 384 pixels in order to reduce the image size by half. Afterwards, the same downscaling and normalizing procedure was performed as for the RT dataset.

We were not able to achieve good results on this task using the machine learning models. We speculate that extra information like temperature and wind speed is needed to do well on this task. Given the limited time frame of this study, we instead focused on forecasting of the real-time radar data. In Figure A.7 you can see an example of the forecasts of the MFBS data.

[1]The MFBS dataset is available at: `https://dataplatform.knmi.nl/dataset/rad-nl25-rac-mfbs-em-5min-netcdf4-2-0`



(a) (b)

Figure A.7: Example of forecasts of the MFBS data. We found the models were not able to do well on this task