Diverse Explorations of Rainfall Nowcasting with TrajGRU:

Mitigating Smoothness and Fading Out Challenges for Longer Lead Times

Yanghuan Zou

Diverse Explorations of Rainfall Nowcasting with TrajGRU:

Mitigating Smoothness and Fading Out Challenges for Longer Lead Times

by

Yanghuan Zou

to obtain the degree of Master of Science at the Delft University of Technology,
to be defended publicly on Tuesday October 31, 2023 at 9:30 AM.

Student Number:	5458463	
Master Programme:	Water Management	
Project Duration:	January, 2023 - October, 2023	
Thesis Committee:	Dr. M.A. (Marc) Schleiss,	TU Delft, dept. Geoscience & Remote Sensing (CiTG)
	Dr. F. (Francesco) Fioranelli,	TU Delft, dept. Microelectronics (EEMCS)
	Dr. R. (Riccardo) Taormina,	TU Delft, dept. Watermanagement (CiTG)

TU Delft

Contents

Preface

This thesis project marks the completion of my master's degree as a water management student at TU Delft. It is definitively a content and unforgettable journey in my lifetime. I am happy to get this opportunity to explore such an interesting and meaningful topic about precipitation nowcasting with machine learning techniques.

This research was done with the great help of my supervisor, Marc Schleiss, my main supervisor and the chair of the committee. It is my luck to have you as my supervisor. I really like our weekly meetings, in which you patiently speculate and discuss the new thoughts with me. Thank you for your constructive feedback and detailed guidance on my work and presentation during these months, helping me progress on what I am weak at. I would like to thank the rest of the committee. To Riccardo, thank you for the suggestion and inspiration concerning the machine learning part to expand my knowledge. Thanks to Francesco for the valuable feedback on presentations and draft reports, standing from different perspectives to evaluate my works and help me wisely and effectively present it.

Lastly, I want to thank my parents and my friends for their constant support and company. To my parents, even though we are far from each other, I am grateful for your constant and supportive trust in all the things and choices I have made, giving me the freedom to explore what I want. To my friends, I feel fortunate to have encountered many lovely friends who share similar interests, values, and outlooks. Thank you for your prompt reminders and suggestions during critical moments. Life is a one-time journey, and I am grateful for this experience.

Yanghuan Zou
Delft, October 2023

Abstract

Machine learning models offer promising potential in precipitation nowcasting. However, a common issue faced by many of these models is the tendency to produce blurry precipitation nowcasts, which are unrealistic. Previous research on the deep learning model - TrajGRU (Shi et al., 2017) indicated that data imbalance in radar images and the double-penalty effect of pixel-wise loss functions are underlying causes for this blurriness.

In this thesis, we continue to explore various approaches to improve the predictive performance of TrajGRU. Our research has first investigated spatially and temporally enhanced loss functions to address the two remaining issues: data imbalance and double penalty. The second part of our research focuses on manifold optimizations within the model network, such as incorporating additional model inputs or increasing batch size, to understand the model's limitations.

Our results reveal that enhanced loss functions did not lead to predictive improvements and even resulted in undesired checkerboard patterns. Changes to the model network make a difference in the image sharpness and predictive rain evolution. Our visual analysis indicates that a larger batch size generates sharper rain field edges; predictions by using multiple parameter groups exhibit more rain dynamics. The incorporation with other transformed datasets introduces finer structures within rain fields. Although the blurriness has not been completely resolved, our study recommended future work can continue exploring the optimization in the TrajGRU network.

List of Figures

List of Tables

1 Introduction

Due to global climate change, the frequency of extreme weather events is increasing. Making urban areas climate-resilient is becoming the common goal of multiple research domains at the moment. Rainfall forecast is a key ingredient of flood forecasting and early warning systems. Precipitation nowcasting is known as the high-resolution forecasting of rainfall and hydrometeors for zero to two hours into the future (Ravuri et al., 2021). It provides prompt warnings for various emergency services, such as urban flood early-warning systems, traffic control, and agriculture, helping each sector to collaborate more effectively and reduce losses.

Most popular forecast applications are made by numerical weather prediction (NWP) models and radar extrapolation-based models. NWP models, such as STEPS and PySTEPS, make predictions based on the physical equations of the atmosphere. Because they take into account various atmospheric processes and relations, NWPs are able to generate highly sharp and reliable forecasts. However, NWPs often yield less accurate forecasts when the lead time is under two hours. This is primarily due to the limited time available for model spin-up and the challenges associated with assimilating non-Gaussian data (Ravuri et al., 2021). As for radar extrapolation-based models, which are based on advecting present rainfall observations along their current trajectories, they can outperform NWPs in the 0-2 hour range. However, they assume that the motion fields are stationary. This assumption renders them unable to capture convection when the rain is chaotic and changes quickly (Prudden et al., 2020).

The physical constraints and underlying assumptions in NWP and extrapolation-based nowcasting hinder models' performance. As the increasing heavy summer precipitation events are of shorter duration and higher intensity, data-driven models can resolve this complex non-linear nature of rainfall information. Unlike traditional approaches, deep learning models operate without making explicit assumptions or relying on complex mathematical equations. Instead, they utilize vast amounts of raw radar information with convolution operations to uncover intricate patterns and relationships within the dataset. Over the past few years, various deep learning methods have been studied and explored in the precipitation nowcasting domain (Shi et al., 2017; Agrawal et al., 2019; Ravuri et al., 2021; Ayzel et al., 2020).

Convolutional neural network (CNN) is considered good at spatial learning (Reichstein, 2019). A typical CNN model related to precipitation nowcasting is U-Net, first used by Agrawal et al. (2019) who trained it directly on one-hour prediction of radar fields. In terms of recurrent neural network (RNN) which is considered better at sequence learning, Shi et al. (2015) first proposed an RNN-based model, the convolutional long short-term memory (ConvLSTM), to capture the spatial and temporal features of radar echo sequence. Shi et al. (2017) later improved it using the trajectory gated recurrent units (TrajGRU) which carries out trajectory convolution between different time steps to learn the structure of spatial variations for recurrent connections. Additionally, deep generative models, such as GANs and DGMR (Elsmann, 2023; Ravuri et al., 2021), began to be widely studied for precipitation nowcasting tasks. The generative approach, aimed to addressing the blurriness in most deep learning models, samples the most appropriate prediction from the distribution of possible futures instead of returning a single best result.

In this thesis, we investigate the application of TrajGRU for precipitation nowcasting in the Netherlands. The Dutch weather is known for its unpredictability, which poses a substantial forecasting challenge. TrajGRU presents a promising solution by facilitating the active learning of location-specific patterns within recurrent connections. This model addresses the spatiotemporal correction challenges that have traditionally been a primary concern in ConvLSTM models (Shi et al., 2017). Before our works, TrajGRU has been customized for the Netherlands by van der Kooij (2021) and Dekker (2022). According to their conclusion and model performance presented in section 1.1, TrajGRU requires more improvement and experiments to achieve more accurate predictions for longer lead times and higher rainfall intensities. This thesis builds upon the work of van der Kooij (2021) and Dekker (2022), with the primary goal of enhancing the predictive performance of TrajGRU.

1.1. Problem statement

Many deep learning models exhibit a common limitation, as they tend to produce smoothed rainfall fields and consistently underestimate high rainfall values at longer lead times. TrajGRU is no different in this regard.

This project began with the objective of exploring TrajGRU to predict heavy summer precipitation in the Netherlands with lead times of up to 100 minutes. van der Kooij (2021) and Dekker (2022) conducted various experiments and work. van der Kooij (2021) explored the effects of training datasets with different sizes. She discovered that the larger dataset with a smaller proportion of heavy events achieved the best scores in Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). In her second experiment, comparing two loss functions, her final analysis concluded that the balanced loss function could outperform the RMSE loss function for higher rainfall intensities (> 5 mm/h). However, these differences are subtle, and the blurriness still persists in the visual results. van der Kooij (2021) also noted that there appears to be a trade-off between performance at high rainfall intensities and performance at low rainfall intensities. The choice of which model performs 'best' depends on how the loss function defines good model performance.

Dekker (2022) conducted an extensive study on the influence of loss functions, following van der Kooij (2021)'s conclusion about establishing optimal guidance for the model. She investigated five different loss functions that emphasize various features in radar images. For instance, Structural Similarity loss (SSIM) focuses on detecting brightness, contrast, and structural similarity between two images. In the end, she found that using alternative loss functions could improve the quality of predicted images for the first few lead times, but the blurriness persists at longer lead times. She also trained the model on rain rate (RR) instead of normalized pixel value (NPV), as RR pertains to the domain of interest and the distribution of RR values places greater emphasis on higher intensity. However, from a visual perspective, the models trained on RR showed less favorable results, as RR will spread out the rain with longer lead times.

Dekker (2022) identified two possible causes of the blurriness in longer lead times. Firstly, TrajGRU often predicts pixel values close to zero, as zero is the most common state in radar images. Bakkay et al. (2022) stated in this study that about 98% of the pixels have no precipitation, and predicting pixel values close to the majority value helps models minimize errors. Consequently, the data imbalance in the radar dataset leads to the fading of rainfall with increasing lead times. The second reason is the location specificity of the loss functions, often regarded as the 'double penalty' issue (Gilleland et al., 2009; Dekker, 2022; Lagerquist and Ebert-Uphoff, 2022). The pixel-wise loss function requires rainfall features predicted at the exact same location as in the observations, which is challenging. If the intensities in predictions are accurate but the rain field shifts in space by just one pixel, it can result in a significant loss (Lagerquist and Ebert-Uphoff, 2022). To minimize the loss, models will blur the high intensity in predictions.

So far, several studies have noticed the data imbalance issue and proposed novel loss functions. Ko et al. (2022) modified the critical success index (CSI) by increasing the CSI score for minor classes; similarly Lin et al. (2018) proposed the focal loss which focuses on challenging examples and down-weights the importance of easy examples during training. To reduce the influence of enormous zero

values in radar images, Bakkay et al. (2022) added a weighting in the loss function in which the loss for each frame is weighted based on its standard deviation. A single frame with more valued pixels could lead to a higher standard deviation and be assigned a larger weight. In addition to modifying the loss function, Yang and Mehrkanoon (2022) trained models with three precipitation datasets, each containing different percentages of rainy pixels. Their results indicated that datasets with a higher proportion of rainy pixels could lead to less blurriness in predictions. Regarding the double penalty issue, Lagerquist and Ebert-Uphoff (2022) explored the effect of neighborhood loss functions which enable models to compare with neighboring pixels and more tolerant of displacement errors.

1.2. Research objectives

As mentioned, two main issues exist in the current model: blurriness and fading out of predictions. The blurriness primarily relates to the loss of distinct and sharp gradients within predicted rain fields and the smooth edges of rainy areas. At longer lead times, all fine structures are lost and fused together. Fading out focuses on the continuous decay of intensity over time, resulting in the underestimation of high rain intensity in the longer lead times. The overall goal of this research is to identify effective solutions that can make prediction more realistic. To be more specific, the objective is to ensure that fine structures and more actual rain evolution can show up in predictions.

The first objective is to tackle the blurriness in predictions. Following van der Kooij (2021) and Dekker (2022), two factors contribute to the blurring: the imbalance in the dataset and the uncertainty with respect to the location of the rain. We aim to understand and evaluate the impact of the data imbalance and the double penalty issue. In this regard, rather than exploring various model architectures, it is more straightforward to consider alternative modified loss functions that emphasize wet pixels or employ location-invariant loss functions taking the surrounding context into account. Since the double penalty hinders the sharpness, this thesis accepts the scenario that the model may predict an incorrect location as long as it produces sharp images.

The second objective is to break the continuity among output frames and introduce more realistic dynamics, particularly rain growth into the predictions. It is essential to investigate the possible reasons for the continuous fading-out pattern of rain intensity observed in current forecasts. In addition to coping with too many zeros and the location specificity of the loss function, the optimization of the model network or hyperparameters will be tested to evaluate their impact on the rain dissipation.

1.3. Research questions

According to the research objectives mentioned above, the research question of this thesis is:

How can TrajGRU be upgraded to mitigate issues related to the prediction blurriness and fading out?

As mentioned before, the blurriness and fading out are probably caused by the data imbalance and the double penalty. To answer this question, three sub-research questions (SQ) assoticated with research objectives have been formulated:

SQ1: Can model performance be improved by applying temporal or spatial weighting strategies which aim at addressing the data imbalance?

SQ2: Is it possible to sharpen the predictions and avoid the double penalty issue by using spatial neighborhood loss functions?

SQ3: Are there specific elements within TrajGRU, such as hyperparameters and model inputs/outputs, that can be adjusted to enhance predictive performance?

To evaluate the predictive performance of models in precipitation nowcasting, our evaluation strategy integrates both visual and numerical evaluations. The visual analysis involves animations, figures, and

magnified images. This approach allows us to gauge the model's predictions from a user's viewpoint. In parallel, we employ a range of quantitative metrics, such as mean error, Gradient Difference Loss (GDL), and Structural Similarity Index (SSIM). These metrics are instrumental in providing a quantitative assessment of the model's performance across varying lead times. The details about the evaluation will be introduced in section 3.6.

The rest of the thesis report is structured as follows: Chapter 2 provides a description of the target model – TrajGRU, including the overview of the model framework and the basic loss function we use in this thesis. Chapter 3 introduces the radar dataset and manifold approaches studied in this thesis. In Chapter 4, the corresponding visual and metric results for each approach are presented and discussed respectively. In the end, conclusions and key findings of the study are given in Chapter 5.

2

TrajGRU Background

Deep learning models have already demonstrated great potential for short-term precipitation forecasting (Shi et al., 2017; Agrawal et al., 2019; Ravuri et al., 2021; Ayzel et al., 2020). The recurrent neural network (RNN) is a special type of network that includes additional connections to enable the passing of information from one timestep to the next within the same layer of the network. GRU (Gated Recurrent Unit) and LSTM (Long Short-Term Memory) are two types of recurrent neural network architectures that enhance the model's ability to capture long-term dependencies in sequential data. Both GRU and LSTM control the flow of information between inputs and outputs by employing gating mechanisms. These gates determine which information from the past is important and should be retained, and which information should be forgotten or updated. This gating mechanism allows models to maintain both short-term and long-term memory for rainfall information. Compared to LSTM, GRU, with fewer parameters, excels in terms of faster training times and lower memory requirements. In this study, our focus is on the TrajGRU model which is introduced by Shi et al. (2017).

In this chapter, the background information of the TrajGRU architecture is in section 2.1. TrajGRU architecture usually consists of multiple TrajGRU units connected in a recurrent manner. A brief description of TrajGRU unit is provided in section 2.2. The principal training loss function is described in section 2.3.

2.1. TrajGRU architecture

The encoder is a downsampling process, in which the traditional convolution is employed before each TrajGRU cell. The convolution operation in the encoder is used to reduce the image size and abstract important features from different perspectives. For example, some resulting feature maps highlight the rain field's boundaries, while others magnify high rainfall rates in the center. This highlighted feature helping the network easier to learn spatial information. In TrajGRU, the encoder first downsamples the image three times and then the forecaster upsamples it again (Figure 2.1). The forecaster operates in the reverse order of the encoder and incorporates deconvolution operations. The upsampling process involves translating the diverse features into a complete image.

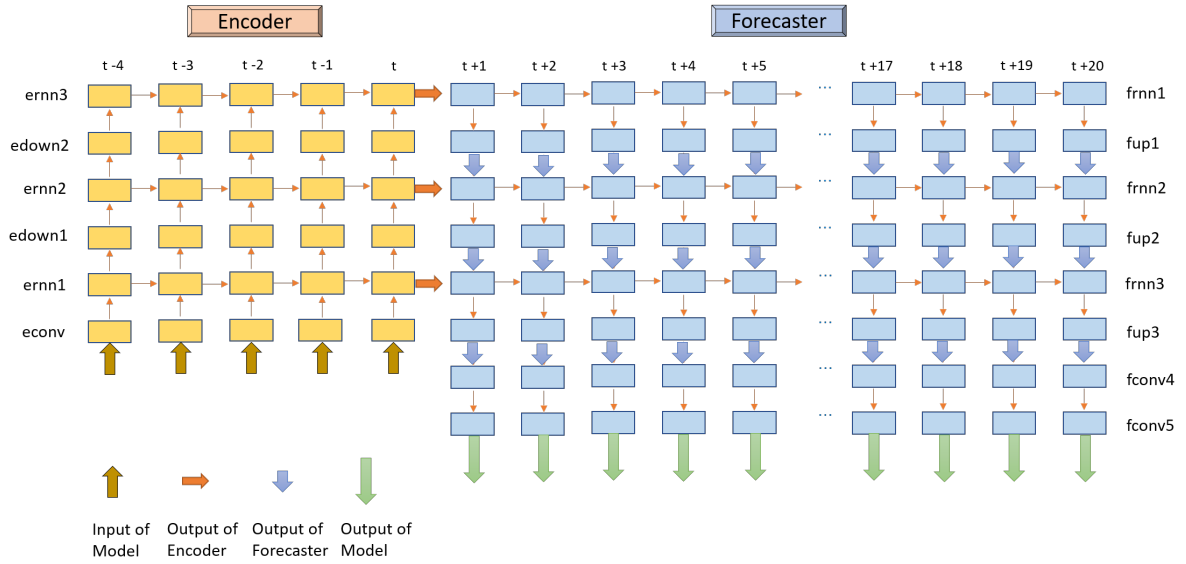

Figure 2.1: The TrajGRU architecture with 5 input frames and 20 output frames; layers named with 'rnn' include TrajGRU unit.

The model architecture in Figure 2.1 indicates that it utilizes the five most recent radar images to generate 20 predictions. The network consists of three main layers, with each layer containing a TrajGRU unit and one convolution (or deconvolution) operation. It's important to note that the outputs of the encoder in the three layers serve as the input for the forecaster. The learnable components during the optimization process are the parameter sets stored in the convolutional filters. Each layer of the encoder or forecaster (Figure 2.1) uses the same set of parameters. Specific details of the parameter settings, such as kernel size and channel number, can be found in Table 2.1. Dekker (2022) mentioned that there are changes in settings when van der Kooij (2021) transferred her code. Although Dekker (2022) recommended switching to the settings from Shi et al. (2017), this thesis employed the same model settings as those used in van der Kooij (2021).

Table 2.1: TrajGRU settings in the benchmark.

Name	Kernel	Stride	Pad	L	Ch I/O	In Res	Out Res	Type	In	In State
econv	7 x 7	5 x 5	1 x 1	-	1/8	480 x 480	96 x 96	Conv	input	-
ernn1	3 x 3	1 x 1	1 x 1	13	8/64	96 x 96	96 x 96	TrajGRU	econv1	-
edown1	5 x 5	3 x 3	1 x 1	-	64/192	96 x 96	32 x 32	Conv	ernn1	-
ernn2	3 x 3	1 x 1	1 x 1	13	192/192	32 x 32	32 x 32	TrajGRU	edown1	-
edown2	3 x 3	2 x 2	1 x 1	-	192/192	32 x 32	16 x 16	Conv	ernn2	-
ernn3	3 x 3	1 x 1	1 x 1	9	192/192	16 x 16	16 x 16	TrajGRU	edown2	-
frnn1	3 x 3	1 x 1	1 x 1	9	192/192	16 x 16	16 x 16	TrajGRU	-	ernn3
fup1	4 x 4	2 x 2	1 x 1	-	192/192	16 x 16	32 x 32	Deconv	frnn1	-
frnn2	3 x 3	1 x 1	1 x 1	13	192/192	32 x 32	32 x 32	TrajGRU	fup1	ernn2
fup2	5 x 5	3 x 3	1 x 1	-	192/64	32 x 32	96 x 96	Deconv	frnn2	-
frnn3	3 x 3	1 x 1	1 x 1	13	64/64	96 x 96	96 x 96	TrajGRU	fup2	ernn1
fup3	7 x 7	5 x 5	1 x 1	-	64/8	96 x 96	480 x 480	Deconv	frnn3	-
fconv4	3 x 3	1 x 1	0 x 0	-	8/8	480 x 480	480 x 480	Conv	fup3	-
fconv5	1 x 1	1 x 1	0 x 0	-	8/1	480 x 480	480 x 480	Conv	fconv4	-

2.2. TrajGRU unit

TrajGRU is an upgraded version of ConvLSTM and ConvGRU that serves as the core of RainGuRu. Instead of solely relying on fixed filters in convolution, TrajGRU employs location-variant filters, increasing the model's flexibility, as discussed in Shi et al. (2017), van der Kooij (2021) and Dekker (2022). In the encoder-forecaster structure, a TrajGRU cell is integrated and positioned in the layer labeled 'rnn' in Figure 2.1.

Zooming in on the typical TrajGRU unit (Figure 2.2), there are two input streams entering each

unit: $\mathbf{H}_{t-1}$, which is the hidden state from the previous lead time, and $\mathbf{X}_t$, which is the input from the original radar image or the previous convolution operation. After processing by gating mechanisms, the output of unit ($\mathbf{H}_t$) becomes the input for the next time step. TrajGRU excels at capturing spatiotemporal correlations compared to other models due to its use of location-variant filters. The flow generator (described in Equation 2.3 in van der Kooij (2021)) is a subnetwork that applies these location-variant filters to determine the flow fields based on the current input and the previous hidden state. These flow fields store dynamically changing local connection structures, which are then incorporated into the reset gate and update gate. The reset gate controls whether the previous state should be cleared or not, while the update gate manages how much of the new information should be written to the state.

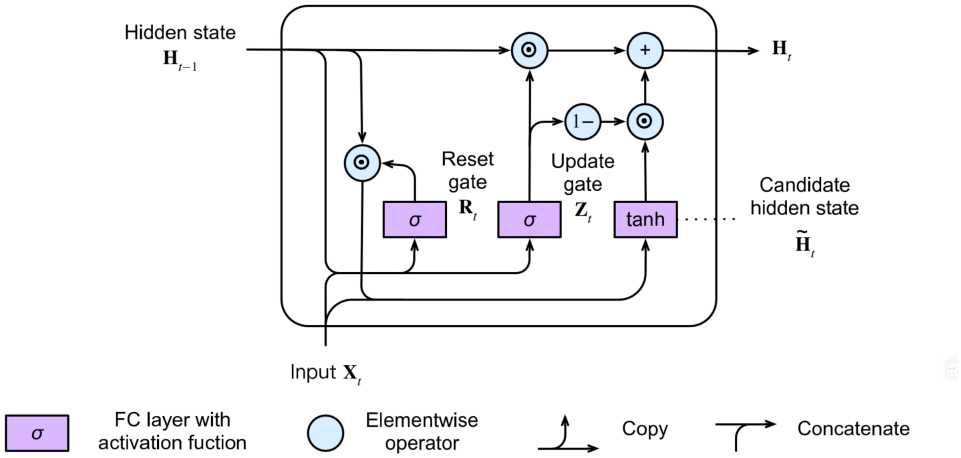

Figure 2.2: A TrajGRU unit illustrating the computational flow. (Source: van der Kooij (2021))

2.3. TrajGRU training loss function

Unless specified otherwise, the default loss function used for training in this thesis is the balanced loss. The balanced loss function is a pixel-wise loss function that takes into account the differences between the predictions and the observations at the pixel level. It is a combination of MAE and MSE (Equation 2.1), with equal weights for both components. The primary distinction between these two metrics is that MSE (Equation 2.2) is more sensitive to outliers than MAE (Equation 2.3) due to the squaring operation. The determination of weights is based on observed rainfall intensity (Equation 2.4). These weights are calculated according to rainfall intensity classes. This weighting strategy rewards the model for correctly predicting heavy precipitation, as discussed in Shi et al. (2017).

$$Loss = BMSE + BMAE \tag{2.1}$$

$$BMSE = \frac{1}{20 \cdot 360 \cdot 360} \sum_{n=1}^{20} \sum_{i=1}^{360} \sum_{j=1}^{360} w_{n,i,j}(F_{n,i,j} - O_{n,i,j})^2 \tag{2.2}$$

$$BMAE = \frac{1}{20 \cdot 360 \cdot 360} \sum_{n=1}^{20} \sum_{i=1}^{360} \sum_{j=1}^{360} w_{n,i,j}|F_{n,i,j} - O_{n,i,j}| \tag{2.3}$$

$$w_{n,i,j} = \begin{cases} 1 & (R_{n,i,j} \leq 1) \\ R_{n,i,j} & (1 \leq R_{n,i,j} \leq 1) \\ 30 & (R_{n,i,j} > 30) \end{cases} \tag{2.4}$$

Where, $F_{n,i,j}$ and $O_{n,i,j}$ are the predicted and observed normalized pixel values, respectively, corresponding to the $(i, j)^{th}$ pixel in the n^{th} frame. The weights $w_{n,i,j}$ in RR domain refer to the rainfall value according to certain categories.

3
Methodology

In this chapter, a lot of different ideas for improving the performance of TrajGRU in terms of the blurriness and fading out patterns have been explored. Section 3.1 provides basic information about the radar dataset and their preprocessing. In section 3.2, some enhanced loss functions to deal with data imbalance and double penalty problems are described. Followed by section 3.3, optimizations of TrajGRU hyperparameters are proposed. In section 3.4, approaches about using transformed radar datasets as new or additional model input are introduced. The verification scores and strategy used to judge the pros/cons of each model are presented in section 3.5. Lastly, the training setting in DelftBlue (the supercomputer device) is briefly described in the section 3.6.

3.1. Data

The Royal Netherlands Meteorological Institute (KNMI) operates two C-band Doppler weather radars situated in Herwijnen and Den Helder (Figure 3.1), which together cover the entire area of the Netherlands. The raw radar composites can be obtained through the KNMI data platform using an API. The archived composites are accessible under the product name '$radar\_tar\_refl\_composites/versions/1.0$' while the near real-time images are available after 3 to 4 minutes and can be accessed under the product name '$radar\_reflectivity\_composites/versions/2.0$' For further information on the radar dataset, please refer to van der Kooij (2021). KNMI produces a radar composite with a spatial resolution of 1×1 km and a temporal resolution of 5 minutes. The data are stored in hdf5 files, containing 2D arrays representing a 700×765 pixel image with 8-bit (0-255) radar reflectivity values. These 8-bit integers can be converted into reflectivity values in dBZ using Equation 3.1. Then, rainfall rates can be derived from radar reflectivity with Equation 3.2. The pixels for which no measurements are available, because they are outside of the range of the radars or the radar was turned off, have a no-data value of 255.

$$Z = (pixel\ value * 0.5) - 32 \tag{3.1}$$

$$z = 10^{Z/10} = 200R^{1.6} \tag{3.2}$$

Where, z means reflectivity in mm^6/m^3; Z is reflectivity in dBZ; R refers to rainfall rates [mm/h]; Dekker (2022)) indicated that the model trained on NPV yields more visually realistic and sensible predictions compared to the model trained on RR. Consequently, in this thesis we continue using NPV as the unit of target data. The relationships between different units can be understood by referring to the equations above.

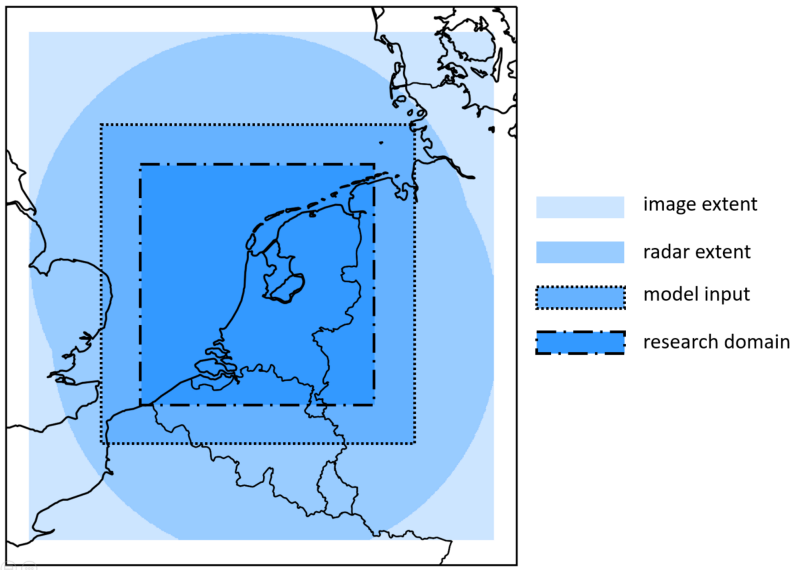

Figure 3.1: Image extent: the 700x765 array the data is provided in. Radar extent: the pixels containing radar measurements. Model input: 480x480 array. Research domain: 360x360 array. (Source: van der Kooij (2021))

The original radar image used as a model input has a fixed size of 480×480 pixels, as shown in Figure 3.1. However, during the loss calculation, the model's performance will be evaluated within a domain of 360×360 pixels, referred to as the 'research domain.' This adjustment is made primarily to mitigate border effects. The model requires the ability to detect rainfall cells approaching in order to make accurate predictions, and this is challenging for pixels near the border (van der Kooij, 2021).

The radar data cover the period from 2008 to 2020. The dataset is divided into three parts: training, validation, and testing. The training period spans from 2008 to 2018; the validation dataset consists of data from 2019, and the remaining data from 2020 are used for testing (Table 3.1).To enable the prediction of heavy rain, van der Kooij (2021) proposed an event extraction strategy that selects sequences containing precipitation. It's important to note that a sequence is defined as a series of 25 consecutive frames. Different extraction strategies will result in different composite sizes. In this thesis, we directly employ radar sequences which have already been pre-processed by van der Kooij (2021). The pickle name including 'c1' (Table 3.1) represents that all images in the dataset have at least 1% of the area with a rain intensity of 1mm/h, and at the same time, more than 12 frames in the sequence contain the peak intensity of 10mm/h. Since the research focus is on heavy summer precipitation events, sequences extracted using stricter criteria are used as summer testing events. Similarly, the name of a pickle file labeled with 'c6' indicates that there is more than 2% rain area with an intensity of 1mm/h, and more than 12 frames with a pixel value of 30mm/h. The resulting number of eligible sequences selected by different strategies is shown in Table 3.1. All experiments are trained and tested with the same dataset.

Table 3.1: Overview of sequences in training, validation, testing datasets

Data types	Year	Number of sequences	File name
Training data	2008-2018	185,324	events_train_all_c1.pickle
Validation data	2019	15,015	events_valid_all_c1.pickle
Testing data	2020	476	events_test_heavy_c6.pickle

3.2. Alternative loss functions

The loss function used in the benchmark is the balanced loss. In this section, the balanced loss is modified by adding additional temporal weights on lead times or spatial weights on each pixel. In addition to that, we also consider the case where the loss function includes several neighborhood pixels instead of the classical pixel-by-pixel comparisons. The objective of modifying the loss function is to help machine learning models extract key information from the training data and reward models that

have desirable properties (e.g., in terms of smoothness, structure or distributions).

3.2.1. Temporal weights on balanced loss

Because of the increasing uncertainty from the rainfall movement and evolution, it is hard for a neural network to predict the accurate track of rains at longer lead times. If the high intensity is forecasted at a slightly wrong location from where it actually occurred. The model will strongly get penalized by this error due to the large differences in rainfall rates between the observations and predictions. TrajGRU tends to spread out the rainfall over large areas and smooth out the spatial details. The uncertainty from the complex and dynamic atmosphere discourages the model from predicting high intensity, small scale features.

A new loss function that reduces the model's sensitivity to displacement errors at longer lead times is proposed. In the original code, the overall balanced loss is calculated by averaging loss values at each lead time. In order to reduce the emphasis on the later lead times, which are harder to predict, a decreasing weighting function is used. Two types of decreasing weights are investigated: linear and exponential. In order to make results comparable, we set the temporal weight of the first lead time to 1, and adjust the slope of the weighting function. The temporal weight details are shown in Figure 3.2 and Table 3.2.

In addition to decreasing weighting pattern, we also did some trials in which we just put more emphasis on the middle lead times (Figure 3.2), since the uncertainty between the input and the middle output is much less than with the longer lead times. Models start to blur the predictions from the first few lead times. It is easier for the model to predict the small scale feature changes for shorter lead times, without creating large error. Therefore, by putting more weight on the middle lead times, we expect the model can provide sharper and more accurate predictions for intermediate lead times.

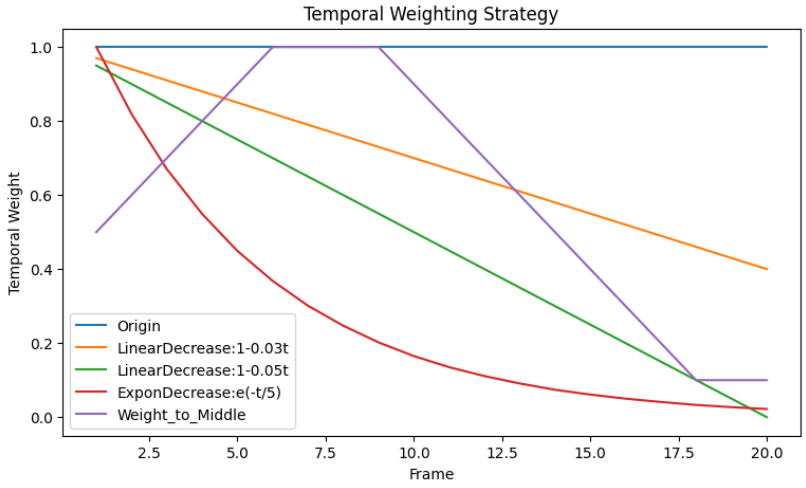

Figure 3.2: Temporal weighting strategies in 20 output frames.

3.2.2. Wet mask

As mentioned in previous works, the blurriness for longer lead times is probably caused by data imbalance in the dataset. In regular radar images, most pixels contain the value of zero. Because of this, the intensity error for the wet pixels only has a negligible effect on overall loss after averaging with more than a hundred thousand of pixels. Too many zeros also result in small and stable loss values during training, which relates to the vanishing gradient problem causing the weights in filters to receive very small updates. Under such a scenario, models make a habit of predicting values close to the majority which is zeros (Dekker, 2022). Few studies already investigated the effect of data imbalance. Bakkay et al. (2022) put more weight on frames with the greater standard deviation by adding additional weights. Ko et al. (2022) used critical success index (CSI) and You et al. (2023) explores the role of the Dice loss function in dealing with sample imbalance.

To mitigate the impact of zeros and only put emphasis on wet regions during training, we integrate a wet mask into the loss computation. Originally, the loss is the average balanced loss across all pixels. However, with the wet mask approach, we first identify "wet pixels" which should be larger than a certain threshold, then assign them a value of 1. This leads to the creation of wet masks for both observations and predictions. Combining these wet masks results in an integrated wet mask where valued pixels indicate rainfall presence in either the Observation or Prediction images. The final loss is then computed by averaging the balanced loss of all wet pixels (Table 3.2). Figure 3.3 explains the concept of applying the wet mask in loss.

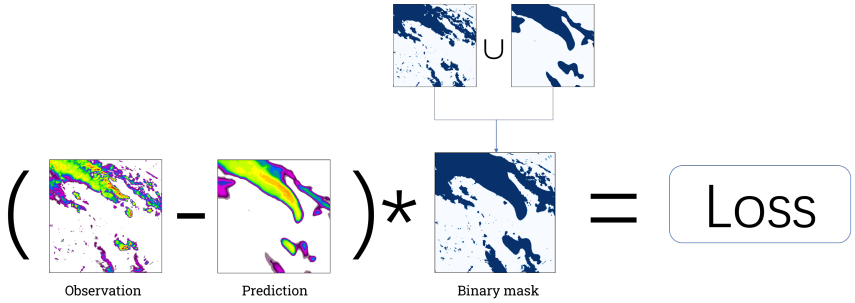

Figure 3.3: The illustration of using wet masks in loss calculation. In the wet mask, wet pixels with a value of 1 indicate that pixel values in observations or predictions exceed the threshold and are included in the optimization process.

3.2.3. Neighborhood loss functions

When addressing the issue of double penalties, employing neighborhood analyses can be a viable option. A neighborhood loss function involves a mean or maximum filter, taken over a square neighborhood of pixels (Lagerquist and Ebert-Uphoff, 2022). Traditionally, pixel-wise loss assesses predictions at pixel C by comparing them with observations at the same pixel location, C. However, pixel-wise verification encourages models to produce blurry predictions (Dekker, 2022). High pixel-wise error will occur if the predicted pixel with high intensity is not situated at the right location. With the neighborhood loss function, models compare target pixels with their surroundings. In this case, models are trained with more freedom and become insensitive to displacement errors. To reduce the blurriness and fading out, this section will use a kernel centered at pixel 'C' to compare the predictions and observations around 'C'. Two types of neighborhood loss functions will be explored and combined into the loss calculation.

Max Pooling

Due to the tendency of high intensity to fade over longer lead times, preventing such underprediction requires a focus on extreme precipitation. Max pooling can enhance the high intensity prediction. It abstracts the highest value in the pixel neighborhood and replaces the center pixel with the max. After the max pooling process, more pixels in feature maps are assigned with high intensity. This approach prioritizes intensity prediction rather than exact location, ensuring high intensities exist in adjacent pixels at least.

In the first phase, both the predictions and observations undergo the max pooling operation before they are compared with each other. The resulting feature maps are then used for balanced loss calculation (Table 3.2). Theoretically, larger kernel sizes perform better at the prediction of extreme precipitation, because the sliding kernel covers a broader pixel range and has greater possibility to access higher rainfall values. Thus, the effect of different kernel sizes is also investigated, with kernel sizes of 2, 7, and 15. The stride size remains fixed at 1.

Table 3.2: Formulae for vairous enhanced loss functions.
Note: n is the index for one of the 20 outputs; w_n is the temporal weight in nth image; $m_{n,i,j}$ is the pixel weight to identify wet or dry pixel (1 or 0); $w_{n,i,j}$ means the spatial weight at pixel *(i, j)* in the nth output frame; $F_{n,i,j}$ represents the forecasted value at pixel *(i, j)* in the nth output frame; $O_{n,i,j}$ is the observed values at pixel *(i, j)* in the nth output frame; r means the applied kernel size; $F^{max}_{n,i,j}(r)$ is the maximum forecast in the neighbour of width r centered at pixel *(i, j)*; $F(r)$ represents all the forecasted values within a neighborhood of width r centered at a specific pixel *(i, j)*

	BMSE	BMAE
Traditional Version	$\frac{1}{20\times360\times360}\sum_{n=1}^{20}\sum_{i=1}^{360}\sum_{j=1}^{360} w_{n,i,j}(F_{n,i,j}-O_{n,i,j})^2$	$\frac{1}{20\times360\times360}\sum_{n=1}^{20}\sum_{i=1}^{360}\sum_{j=1}^{360} w_{n,i,j}\lvert F_{n,i,j}-O_{n,i,j}\rvert$
Temporal Weight Version	$\frac{1}{20\times360\times360}\sum_{n=1}^{20} w_n\sum_{i=1}^{360}\sum_{j=1}^{360} w_{n,i,j}(F_{n,i,j}-O_{n,i,j})^2$	$\frac{1}{20\times360\times360}\sum_{n=1}^{20} w_n\sum_{i=1}^{360}\sum_{j=1}^{360} w_{n,i,j}\lvert F_{n,i,j}-O_{n,i,j}\rvert$
Wet Mask Version	$\frac{1}{20\times360\times360}\sum_{n=1}^{20}\sum_{i=1}^{360}\sum_{j=1}^{360} m_{n,i,j}w_{n,i,j}(F_{n,i,j}-O_{n,i,j})^2$	$\frac{1}{20\times360\times360}\sum_{n=1}^{20}\sum_{i=1}^{360}\sum_{j=1}^{360} m_{n,i,j}w_{n,i,j}\lvert F_{n,i,j}-O_{n,i,j}\rvert$
Max-pooling Version	$\frac{1}{20\times360\times360}\sum_{n=1}^{20}\sum_{i=1}^{360}\sum_{j=1}^{360} w_{n,i,j}\left(F^{max}_{n,i,j}(r)-O^{max}_{n,i,j}(r)\right)^2$	$\frac{1}{20\times360\times360}\sum_{n=1}^{20}\sum_{i=1}^{360}\sum_{j=1}^{360} w_{n,i,j}\lvert F^{max}_{n,i,j}(r)-O^{max}_{n,i,j}(r)\rvert$
Prediction-oriented Version	$\frac{1}{20\times360\times360}\sum_{n=1}^{20}\sum_{i=1}^{360}\sum_{j=1}^{360} w_{n,i,j}(F_{n,i,j}(r)-O_{n,i,j})^2_{min}$	$\frac{1}{20\times360\times360}\sum_{n=1}^{20}\sum_{i=1}^{360}\sum_{j=1}^{360} w_{n,i,j}\lvert F_{n,i,j}(r)-O_{n,i,j}\rvert_{min}$

Observation-oriented evaluation

Similar to the max pooling approach, the model continues using the sliding kernel over prediction and observation images. Instead of comparing the max value in the target neighborhood, in this section, models use a patch centered at pixel C to compare the predictions and observations around C, and calculate the minimum difference. There are two types of evaluation settings: prediction-oriented and observation-oriented. In the observation-oriented approach, if the patch is 3 x 3, the center pixel in the observation patch is compared with 9 neighboring pixels in the corresponding position in predictions (Figure 3.4). Later, the minimum difference between the center pixel and neighboring pixels will be considered in the loss calculation. Related formulas about neighborhood loss functions are shown in Table 3.2. By using the smallest difference, the model is expected to minimize the high displacement error and mitigate the double-penalty problem. The effect of two types of evaluation settings will be studied.

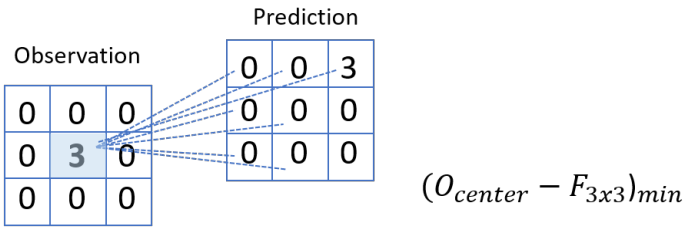

Figure 3.4: Observation-oriented evaluation, in which the central pixel in the observation should be compared with all predicted pixels in the same neighborhood. The minimum difference among them will be considered in the loss evaluation. (In prediction-oriented evaluation, the central pixel is in the prediction.)

3.3. Hyperparameter tuning

In previous research conducted by van der Kooij (2021) and Dekker (2022), their models were trained on a personal workstation and the VR-Lab infrastructure. In contrast, models in this thesis were trained on DelftBlue, a high-performance supercomputer, rather than a personal computer. The difference between these devices is that DelftBlue could offer higher performance and larger memory, such as RAM in a single GPU node. Under this new condition, some hyperparameters can be adjusted to a larger value,

such as batch size and learning rate. Following, the effect of three types of alterations in TrajGRU will be studied. First, using the larger batch size and learning rate is introduced in section 3.3.1; then changes of filter number are described in section 3.3.2; lastly, the increase of parameter groups in TrajGRU is discussed in section 3.3.3.

3.3.1. Batch size and learning rate

Hyperparameters of models will affect the performance of the network along its time to convergence. Batch size is a crucial hyperparameter in modern deep learning systems. Batch size refers to the number of training sequences utilized in the gradient estimation process in each iteration. Understanding its impact is challenging (Balles et al., 2017; Smith et al., 2018; Zhang et al., 2019), as its effects can vary across different cases and models (Shallue et al., 2019). In general, larger batch sizes yield better gradient estimates and lead to shorter training times by reducing the number of required steps (Hoffer et al., 2018; Zhang et al., 2019). However, too large of a batch size will lead to poor generalization (Keskar et al., 2017; De et al., 2017). On the other hand, some stated that a small batch size can converge faster and predict better than the large batch sizes, as the noise helps the model escape from the 'sharp minima' (Keskar et al., 2017; Kandel and Castelli, 2020) to reach a more global minima.

Using larger batch size has another advantage is that the entire data-set can be fully utilized. In previous works, the loss value starts to become stable and reaches the minimum after around 30k iterations. In this case, more than half of sequences in the dataset still not get trained. Utilizing all data helps the model generalize better to unseen data. It learns patterns from the entire dataset, improving its ability to make accurate predictions on new, unseen data. When using a larger batch size, each iteration is able to involve 8 or 10 sets of sequence, meaning the entire data-set can be fully used.

The batch size in the previous works is fixed to 2 because of the limited GPU RAM. As the model is trained on DelftBlue, larger available GPU RAM memory is accessible. The maximum batch size can be up to 10. Kandel and Castelli (2020) suggests that the number of batch sizes should be a power of 2 to take full advantage of GPUs processing. Thus, the batch size of 8 are used in our models. Generally, larger batch sizes require larger learning rates (Balles et al., 2017). The experiments with different combinations of batch size and learning rate are shown below in Table 3.3.

Table 3.3: The overview of experiments with different combinations between batch size and learning rate.

Experiments	Batch Size	Learning Rate
Benchmark	2	0.0001
BS_8_1	8	0.0001
BS_8_2	8	0.0002
BS_8_4	8	0.0004
BS_8_5	8	0.0005
BS_8_10	8	0.0010

3.3.2. The number of filters

TrajGRU employs an encoder-forecaster structure. In the encoder, the radar image will be down-sampled three times. Downsampling refers to the process of reducing the size of images, which can help reduce computational time and memory usage. However, due to the resizing of images, downsampling accompanies the loss of detailed information. In order to pass on adequate information to the next layer, the key information will be interpreted and be transferred to feature maps after each convolution. In total, there are around 6.1 millions filters in original TrajGRU model.

Filters consist of learnable parameters that are adjusted during training. Compared to the great number of filters we point out above, the number of parameters is much more than that. In general, the great amount of parameters ensure the model ability to capture essential patterns. But, the excessive number of parameters will lead to the issue of model equifinality which means many different parameter sets may perform similarly and yield equally good results. The more number of parameters, the high equifinality. Moreover, excessive parameters attempt to compensate for er-

rors in other parameters and adapt to noise in the dataset, contributing minimally to overall performance.

Our current visual results demonstrate a similar pattern in work with different loss functions or parameter sets, which means the existence of model equifinality. To avoid model equifinality, reducing the number of filters is the idea to make the model simpler and focus on the key information in the dataset. There are three modes chosen to be presented (Table 3.4), with decreasing number of filters: Light, Medium, and Extreme. The extreme mode with the least number of filters is going to understand whether the neural network will suffer a breakdown. The input channel before starting the encoder is always 1. The amount of output channels represents the number of resulting feature maps, determined by filter number. The number of channels in forecaster will be changed correspondingly.

Table 3.4: The overview of experiments with decreased channel number in encoder.

	Channel Input / Channel Output			
Layer Name	**Benchmark**	**Light**	**Medium**	**Extreme**
econv	1/8	1/8	1/4	1/4
ernn1	8/64	8/64	4/8	4/4
edown1	64/192	64/64	8/16	4/4
ernn2	192/192	64/64	16/16	4/4
edown2	192/192	64/64	16/16	4/4
ernn3	192/192	64/64	16/16	4/4

3.3.3. Multiple parameter groups

Upon observing the predicted results, it's interesting to note that 20 predicted frames exhibit a consistent spatial pattern within the wet region. The only difference in spatial details between lead times is that forecasts for later lead times are blurred versions of the first lead time (Figure 4.2). This predictive feature indicates the predictability limit of TrajGRU. On the other hand, an intuitively irrational setting in the TrajGRU architecture is found, where all images in the encoder or forecaster share an identical parameter group. Examination of parameter settings in Table 2.1 reveals that, within the same layer, images at different lead times undergo identical convolution operations. The encoder processes 5 input images, while the forecaster manages 20 predicted outputs. Visual analysis suggests that the consistent pattern may originate from TrajGRU using a single parameter set. During the training, learnable parameters are updated per iteration. Compared with last lead times, the adjustment of parameters largely depends on performance at first lead times, because it is easier for models to realize the correct prediction and reduce the overall loss value. On the contrary, there is more uncertainty and changes in the last few lead times, requiring more radical adjustment of parameters. In this case, parameters stored in filters contain information which are more associated with how to reproduce the prediction at the first lead time.

The main goal of this section is to improve the model's ability to predict unseen patterns. To break the continuous fading out pattern and reduce the continuity between the lead times, a strategy of multiple parameter sets is implemented here. This experiment intends to first understand the effect of using multiple parameter groups. Assigning a specific parameter group to each lead time is quite complicated and time-consuming. To simplify the process, 20 outputs are split into 4 groups, thereby increasing the number of parameter sets from 1 to 4. The uncertainty in precipitation motion increases with lead time. This requires an increase in the learning rate for later lead times. This section will also explore the influence of varied learning rates across different parameter groups. Details of experiments are shown in Table 3.5.

Table 3.5: The overview of experiments with 4 sets of parameter in forecaster.

	Learning Rate			
Job Name	**[5-25 min]**	**[30-50 min]**	**[55-75 min]**	**[80-100 min]**
PG1	1	1	1	1
PG2	1	2	3	4
PG3	1	5	10	15

3.4. Changes in model inputs and outputs

In the standard procedure, only raw radar images are used in the model. TrajGRU might not obtain sufficient information regarding the spatial structure and temporal variation of rain, resulting in fading out and the blurriness in predictions. Moreover, models relying on a single input source can be sensitive to inherent noise within that data. Incorporating multiple data sources can help alleviate this issue and offer a more balanced and robust perspective of underlying patterns. This section aims to incorporate various modified data types and assesses their potential in enhancing the model's performance across different aspects. The initial phase will investigate the inclusion of time-differenced rainfall as new model input and output, followed by some trials conduct the utilization of multiple model inputs.

3.4.1. Temporal difference of rain intensity

The model tends to fade out precipitation and does not show any signs of predicting rain growth in later lead times. The raw radar image, which serves as the model's input, represents real-time precipitation. In this case, the model needs to learn and understand rain growth and decay from sequential real-time radar images. By subtracting two consecutive radar images, we can obtain time-differenced data that directly represents rain growth or decay over a 5-minute period. In this experiment, the radar image, which is the current model input, will be replaced by time-difference data. $\mathbf{dR} = \mathbf{R}_t - \mathbf{R}_{t-1}$. $\mathbf{R}_t$ is the radar image; **dR** is the time-differenced image. Note that the distribution of time-differenced data becomes symmetric (Figure 3.6). The intention here is to provide the model with direct information about temporal variations, facilitating its prompt utilization and learning from the given data.

The Figure 3.5 illustrates the transition of the model input from radar data to time-differenced data. As for time-differenced images, there are pixels with negative values that represent the rain decay. It's important to note that the original model had 5 inputs. Using time-differenced data as the model input reduces the number of input images to 4. In addition, as the model output is time-differenced, in order to obtain the target rainfall intensity, the radar image at the present moment should be added with time-differenced output.

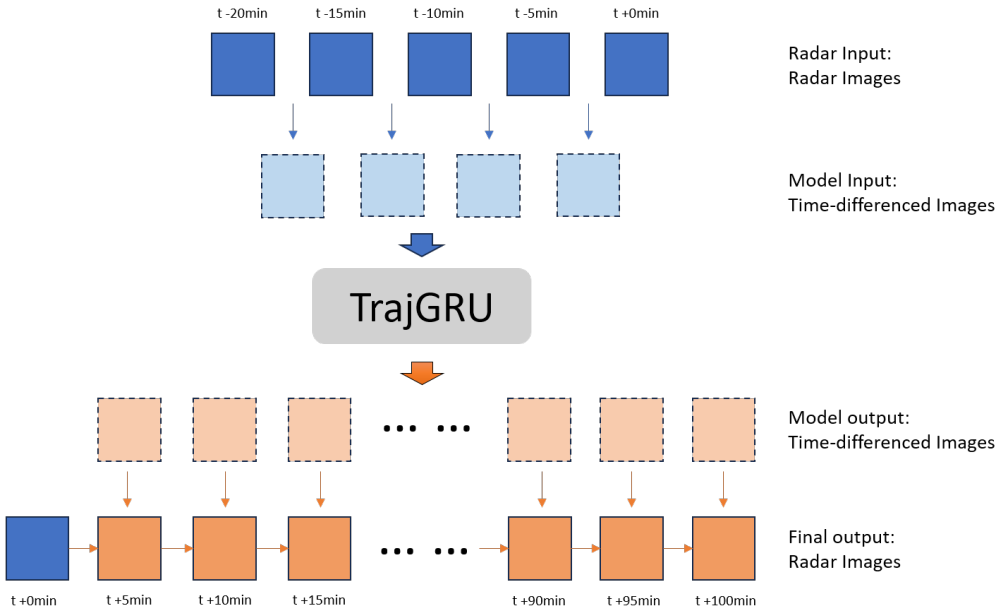

Figure 3.5: The transition from original radar images to time-differenced images; Note: the input number decreases from 5 to 4, and the output number is still 20.

In this thesis, two trails related to the time-differenced input are explored. The only difference between the two experiments lies in the weighting strategy: one follows the original strategy putting more weight on high rain rates, while the other employs the absolute value of the time difference for each pixel to transfer the emphasis to the maximum temporal changes.

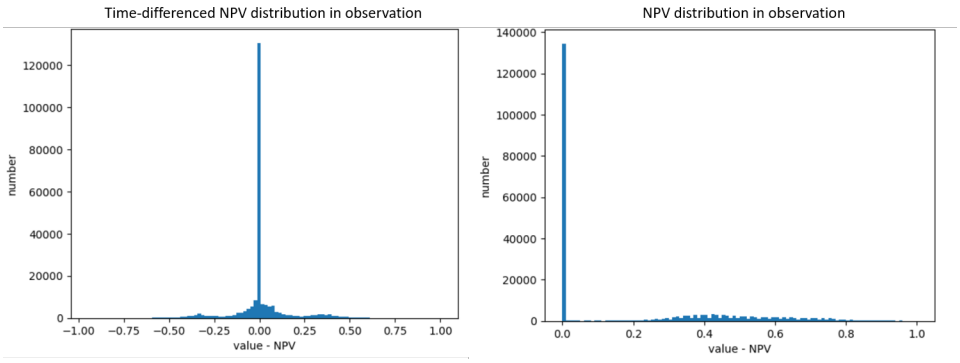

Figure 3.6: The distribution of the training data in NPV and time-differenced NPV.

3.4.2. Multiple model inputs

In addition to using time-differenced data as model input, there are various transformations to radar images that can be explored. Dekker (2022) conducted a experiment on the effect of different data units, comparing NPV (applied domain) and RR (target domain). It was observed that there is a significant imbalance in the occurrence frequency of different precipitation values, resulting in different emphasis. This difference in distributions of two datasets can affect the model's focus, for example models trained on NPV will particularly focus on the edges and low rainfall rates (Figure 3.7), while model trained on RR did in an opposite way. This phenomenon is caused by the gradient difference. The range from 0 - 0.1 mm/h is transformed to 0 - 0.42 NPV, enlarging its gradient in NPV domain. Dekker (2022) recommended in her thesis that switching to the rainfall rates is preferable, as the way we perceive an image depends on the relation between rain rates in an image. It is worth exploring a combination of both domains to get the details in both boundary and within the rain fields.

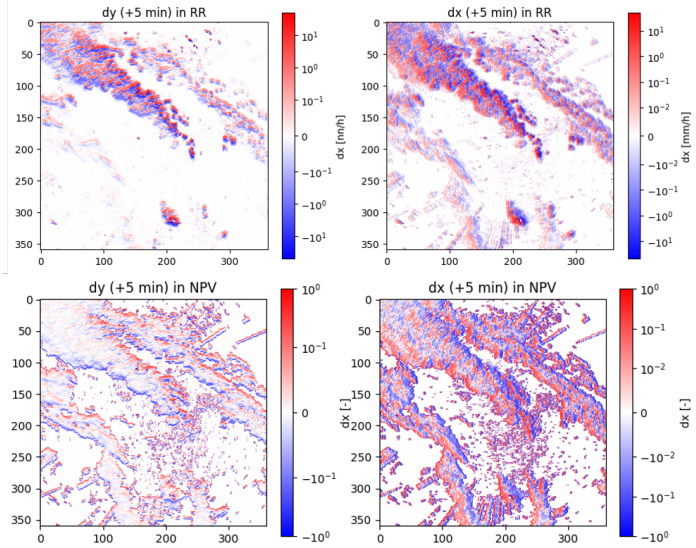

Figure 3.7: The gradient maps in the y or x direction, in both NPV and RR domain.

By incorporating multiple sources of data, models are expected to capture various aspects of the underlying patterns and relationships at the same time, and gain a more comprehensive understanding of rain evolution. In the study by Dekker (2022), RR images serve as both model input and output. In this thesis, model inputs incorporate both RR images and NPV images, while the output remains as NPV images. Time-differenced images are combined with the original radar image to encourage models to capture temporal changes (Figure 3.8). Gradient images in the y-direction are another transformation, used to prompt models to learn gradients between pixels and sharpen predictions. Additionally, radar images with pixels filtering NPV smaller than 0.854 are incorporated in the model inputs to improve the high-intensity prediction.

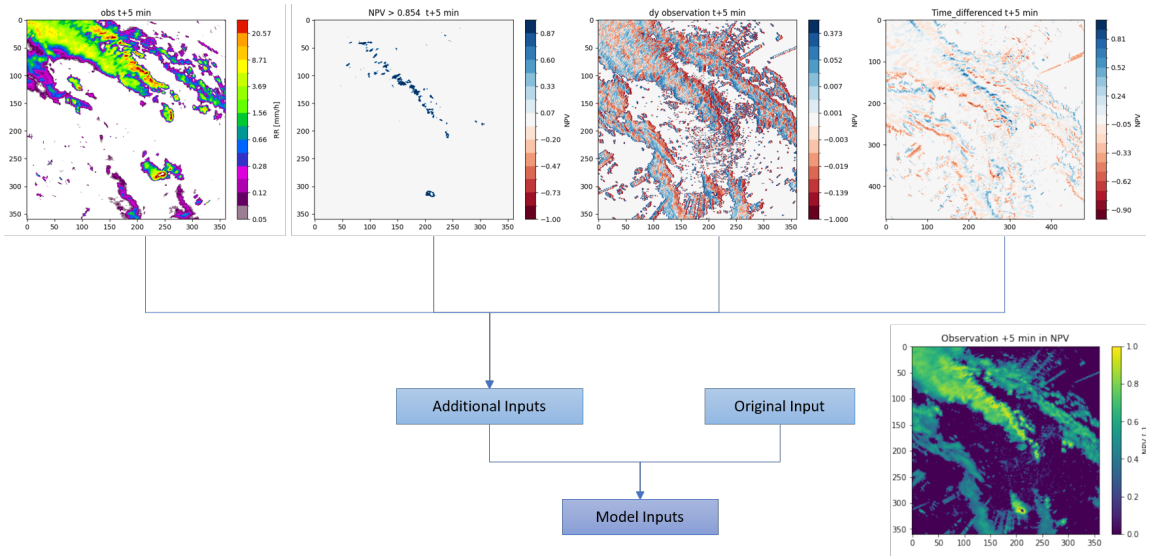

Figure 3.8: The brief illustration of experiments with additional model inputs.

3.5. Verification

This section introduces the verification method used for the outcome. As mentioned before, the benchmark in this thesis is van der Kooij (2021)'s model trained on DelftBlue. The assessment of the predicted performance is based on both visual analysis and several metrics. Each regular metric biases a different aspect in the prediction. Since our analysis object is the radar image which will be displayed on forecast applications, visual analysis is prioritized in this regard. Ravuri et al. (2021) stated that the same prediction can be judged as significantly different by expert meteorologists, but the scores do not provide this insight. Our thesis also highlights the limitation of using existing metrics to evaluate forecasts when using time-differenced as model output, which is discussed in section 4.4.1

3.5.1. Visual analysis

Visual analysis consists of three types: regular prediction images, animation and zoomed-in images. The visualized predictions will provide us with the initial impression of the model's performance. Five real precipitation events have been selected for visual analysis. In addition to these actual events, some artificial events are created to gain a better understanding of the model's predictive patterns. Precipitation events are diverse, dynamic and complicated. For each single precipitation event, features such as the intensity, movement, and structure of a precipitation system can change randomly over time. Analyzing and describing these dynamic changes to evaluate the prediction quality is challenging, as there are too many aspects can be considered and they are inherently linked to each other. To focus on one specific feature, for example heavy shower movements, we can create some artificial events to simplify and emphasize the feature. This artificial events help readers to observe and draw effective conclusions. The most commonly used sequence is depicted in Figure 3.9. The large green patch represents the main rain field, while the small red patch within it indicates an isolated shower with a high rain rate. Both patches move at a certain pace, but in different directions. The small patch moves from the upper left to the bottom right, while the large patch moves in the opposite direction.

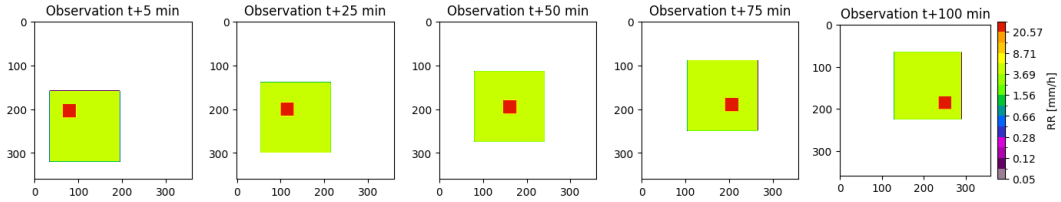

Figure 3.9: Artificial radar sequence help better analysis and comparison between observations and predictions by simplifying the precipitation features such as the distribution and movement patterns of fine structures.

The above method belongs to the static analysis. Although it exhibits predicted images at different lead times, we still need to envisage how the rain moves or how the rain decays over these 100 minutes by ourself. Therefore, GIFs are generated for each experiment evaluation. These animations allow continuous observation of the rain evolution, underlining the dynamic process. The benefits of animations as an analytical tool will be fully reflected in section 4.

3.5.2. Metric analysis

The metrics employed in this thesis are same with Dekker (2022)'s previous work. Continuous metrics, including MAE, MSE, SSIM, MS-SSIM, GDL, Wasserstein, and FFL loss, are plotted across lead times to analyze model performance in temporal terms. Categorical metrics (MSE, MAE, CSI, and Frequency bias) assess accuracy across various rain intensity categories, enabling an analysis of their impact on the prediction of distinct rain intensities (mm/h). The used categories are: [0, 0.1], (0.1, 0.5], (0.5, 1], (1, 5], (5, 10], (10, 20], (20, 30], and > 30. Below is the brief introduction of each metric; more comprehensive information can be found in Dekker (2022).

Mean Absolute Error (MAE): Measures the magnitude of the absolute error. A smaller MAE means better performance. (The definitions of certain variables can be found in section 3.2.3.)

$$\mathrm{MAE} = \frac{1}{20 \cdot 360 \cdot 360} \sum_{n=1}^{20} \sum_{i=1}^{360} \sum_{j=1}^{360} |F_{n,i,j} - O_{n,i,j}|$$

Mean Squared Error (MSE): Measures the magnitude of the squared error. It punishes large errors to a greater extent than the MSE. The lower the better.

$$\mathrm{MSE} = \frac{1}{20 \cdot 360 \cdot 360} \sum_{n=1}^{20} \sum_{i=1}^{360} \sum_{j=1}^{360} (F_{n,i,j} - O_{n,i,j})^2$$

Structural Similarity (SSIM) loss: The SSIM looks at the differences in brightness *l(F,O)* , contrast *c(F,O)* and structure similarity *s(F,O)* between two images. μ_F and μ_O are the means of observations and predictions. σ_F and σ_O represent the standard deviations of observations and predictions.C_1, C_2 and C_3 ($C_3 = C_2/2$) are small positive constants to avoid division by zero and to obtain numerical stability. The value of SSIM ranges between 0 and 1, with 1 indicating perfect similarity.

$$\mathrm{SSIM}(F,O) = l(F,O) \cdot c(F,O) \cdot s(F,O) = \left(\frac{2\mu_F\mu_O + C_1}{\mu_F^2 + \mu_O^2 + C_1}\right) \cdot \left(\frac{2\sigma_F\sigma_O + C_2}{\sigma_F^2 + \sigma_O^2 + C_2}\right) \cdot \left(\frac{\sigma_{FO} + C_3}{\sigma_F\sigma_O + C_3}\right) = \frac{(2\mu_F\mu_O + C_1)(2\sigma_{FO} + C_2)}{(\mu_F^2 + \mu_O^2 + C_1)(\sigma_F^2 + \sigma_O^2 + C_2)}$$

$$\mathcal{L}_{\mathrm{SSIM}}(F,O) = 1 - \frac{1}{20 \cdot 360 \cdot 360} \sum_{n=1}^{20} \sum_{i=1}^{360} \sum_{j=1}^{360} \mathrm{SSIM}(F,O)$$

Gradient Difference Loss (GDL): This loss penalises gradient differences between the prediction and the observations by only considering the neighbour pixel intensity differences. α is an integer of 1 or larger.

$$L_{GDL}(F,O) = \frac{1}{20 \cdot 360 \cdot 360} \sum_{n=1}^{20} \sum_{i=1}^{360} \sum_{j=1}^{360} \left\| O_{n,i+1,j} - O_{n,i-1,j}| - |F_{n,i+1,j} - F_{n,i-1,j} \right\|^{\alpha} + \\ \left\| |O_{n,i,j-1} - O_{n,i,j+1}| - |F_{n,i,j-1} - F_{n,i,j+1} \right\|^{\alpha}$$

Wasserstein loss: The idea behind this loss is that it sees every pixel value as a mass at a location and that it calculates how much it would cost to transfer these masses to the correct pixel to match the observation. The parameter p determines the power to which the euclidean distance, between location x and y, is taken

$$W(O,F) = \min_{\gamma \in \mathbb{R}} \frac{1}{20 \cdot 360 \cdot 360} \sum_{n=1}^{20} \sum_{i=1}^{360} \sum_{j=1}^{360} \gamma_{n,i,j} \frac{1}{p} ||x_{n,i} - y_{n,j}||_p^p$$

3.6. Model training

All experiments were trained on a new device - DelftBlue. DelftBlue is the supercomputer at TU Delft, which can offer higher performance and abundant computing resources. There are 10 GPU nodes, with each node having 4 GPUs: NVIDIA Tesla V100S 32GB. The RAM (Random-access memory) of each GPU node can go up to 256GB. In DelftBlue, because of easier access to the larger memory and more GPU resources, we can explore the potential of various configuration settings through different combinations, such as larger batch size. The relative disadvantage of training on DelftBlue is the waiting time is uncertain, depending on how much users and jobs are waiting in the queue at the moment.

The benchmark in this thesis was trained on DelftBlue with the identical configuration settings, using the identical configuration settings as those employed in van der Kooij (2021) and Dekker (2022); the only difference is the device. Experiments are trained with the batch size of 2 or 8 depending on the approach, for 100,000 iterations. The learning rate schedule with step-decay, with a decay-factor of LR*0.1 at the 30,000th and 60,000th iteration. The optimization algorithm is Adam. The benchmark works with NPV. The actual and normal training time last for about 1 and a half day.

Table 3.6: An overview of the above mentioned approaches. We come up with these approaches from different perspective, all aimed at mitigating blurriness and fading issues in our current model. Experiments are divided into three main categories based on which component in TrajGRU we make changes.

Category	Approach	Explanation
Loss function	Temporal weight	New loss functions assign specific temporal weights to each output frame.
	Wet mask	A binary mask is applied, and models are optimized by calculating the loss of wet pixels.
	Max pooling loss	The new loss including the max pooling operation focuses on the error of the maximum in each neighborhood.
	Prediction-oriented loss	The new loss compare the central-predicted pixel value with neighboring-observed pixel values to reduce the displacement error.
Model hyper-parameter	Batch size and learning rate	Increased batch size and learning rate.
	Number of filter	Smaller size of the model network.
	Multiple parameter groups	The parameter set among output is split into groups of four.
Model inputs and outputs	Time-differenced dataset	Time-difference of rain intensity as the new model input and output
	Additional model inputs	Various transformed radar images such as gradient images and RR images are used as additional model inputs

4

Results

In this chapter, we present and discuss the results of different approaches, organized into three sections based on the modifications made to TrajGRU. Section 4.1 assesses the effectiveness of the modified loss functions; while Section 4.2 looks at the influence of hyperparameters on predictive performance. The effect of using temporal data or additional transformed inputs are analyzed in Section 4.3. A subset of metric results with lead time is provided in Appendix A. You can also refer to online datasets which stores all visual results: 1) DOI: 10.4121/12437ba3-4cf4-47c4-928b-94dc9bdec663; 2) Miro: `https://miro.com/app/board/uXjVM_zMm0s=/?share_link_id=288593714945`

4.1. Predictive features of the benchmark

Figure 4.1 shows the prediction produced by the benchmark. By comparing these predictions with the ground truth (Figure 3.9), we can identify predictive features in the current model. In terms of the main rain field (green patch in Figure 4.1), its location is properly predicted and consistent with the ground truth. Despite the rectangular shape we make is apparently unrealistic, the shape of the wet region in our artificial sequence is also well-preserved by the model. Concerning the rain intensity, the benchmark tends to overpredict it over the primary rain field in longer lead times, as there is no observed increase in intensity in the ground truth.

Regarding isolated showers (red patch in Figure 4.1) within the rain field, the visual results reveal that the benchmark struggles to predict the motion trajectory of heavy rainfall. In the ground truth, the intensity of these isolated showers remains constant over time. However, the benchmark fails to preserve the high intensity and square pattern, instead it fades out and merges them with their surroundings as time progresses. By summarizing the predictive features of the model, we can conclude that the model provides decent predictions at the global scale but still struggles with predicting internal details within rain fields.

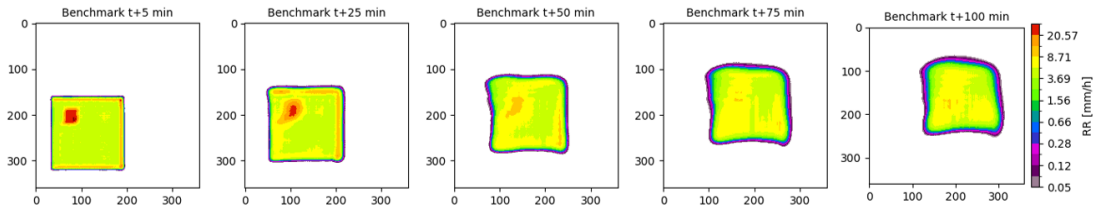

Figure 4.1: The prediction of an artificial event, illustrating current predictive features of the benchmark setting such as continuous fading out of high intensity and blurring pattern for fine structures within rain fields.

4.2. The effect of modified loss functions

4.2.1. Consequences of temporal weights

Some examples of predictions for four different temporal weights are depicted in Figure 4.2. Overall, the new predictions look similar to the benchmark in terms of shape, location and spatial structure

inside the rainy area. Most importantly, there is no clear improvement in sharpness. The models with the new weights still smooth out the rain a lot and appear unable to predict detailed spatial features within the rain fields, especially at the longer lead times. Although fading out still persists, the addition of temporally dependent weights can slow down the rate of decay of high rainfall intensities (e.g., see the red patch in Figure 4.2).

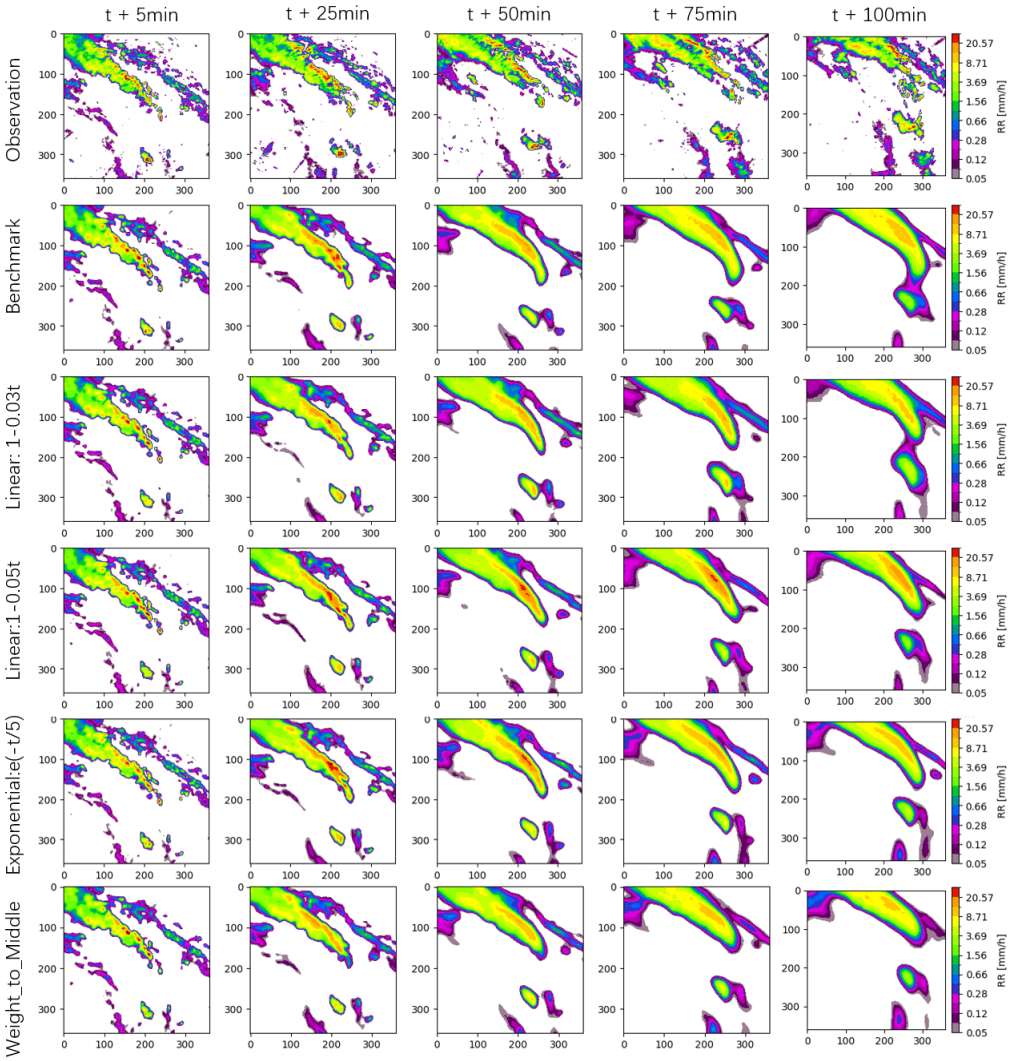

Figure 4.2: The predictions are tested with temporal weighting strategies.

The performance metrics in Figure 4.3 show that, on average, the benchmark performed the best. The experiment in which the weights decrease with lead time shows a slight decrease in all metrics for the first lead time. However, beyond that, the benchmark consistently outperformed the alternative models across all metrics, especially at longer lead times. In the context of emphasizing intermediate lead times, the metrics revealed a certain decrease in values around the 4, 5 and 6 frames. Nevertheless, this weighting strategy lead to the poorest model performance in both the initial and final lead times.

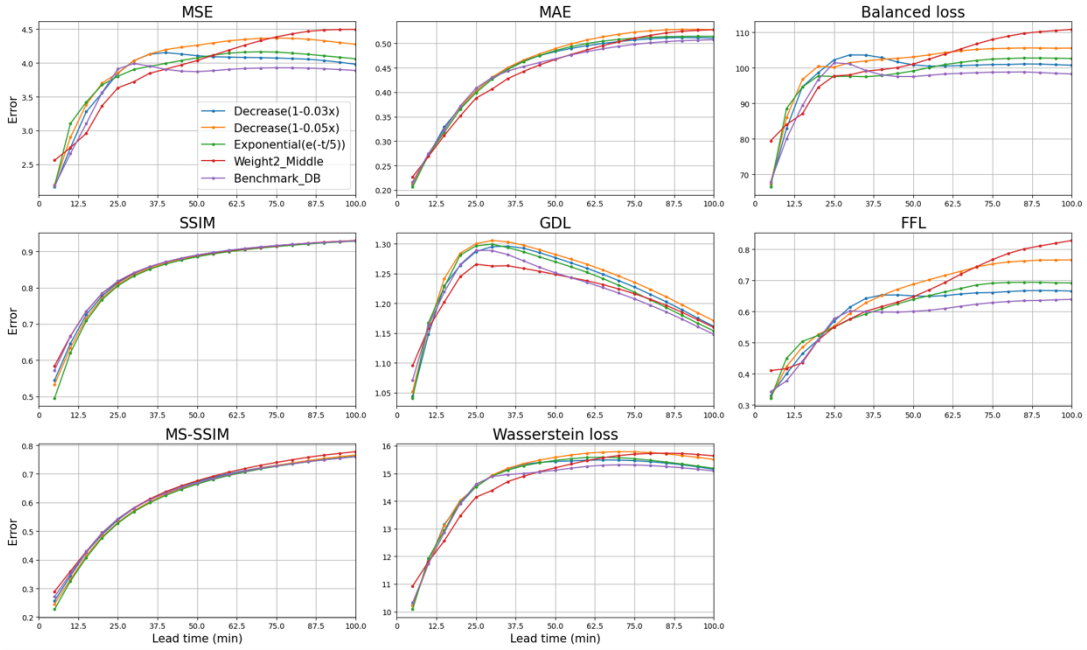

Figure 4.3: Metrics over the lead times for experiments combined with temporal weighting strategies. A lower value indicates a better score for all metrics.

4.2.2. Consequences of wet mask

In this experiment, the loss value is only computed over wet pixels, by excluding those pixels with values smaller than a certain threshold in both observation and prediction. The idea behind using a wet mask is to reduce data imbalance between wet and dry pixels in the training data, and transfer more emphasis on high intensities.

The visual results indicate that models trained with a wet mask struggle to preserve extreme rain pixels over time (Figure 4.4). They consistently display a blurry pattern in longer lead times, similar to the benchmark. When a large threshold is applied, these models fail to predict dry pixels and mistakenly classify them as wet pixels. This is due to the fact that we do not consider these dry pixels in the loss calculation. Because assigning values to them does not penalize the model, the model are free to on dry pixels. Employing a high threshold can enhance performance in the initial lead time (Figure 4.4). When zooming into the circled area in Figure 4.4, more separate small rain fields outside the main field are successfully predicted,which the benchmark fails to predict.

The fact that the wet mask did not produce the expected results indicates that too many zeros in the training data might not be the main reason for the blurriness and decay of precipitation in the longer lead times.

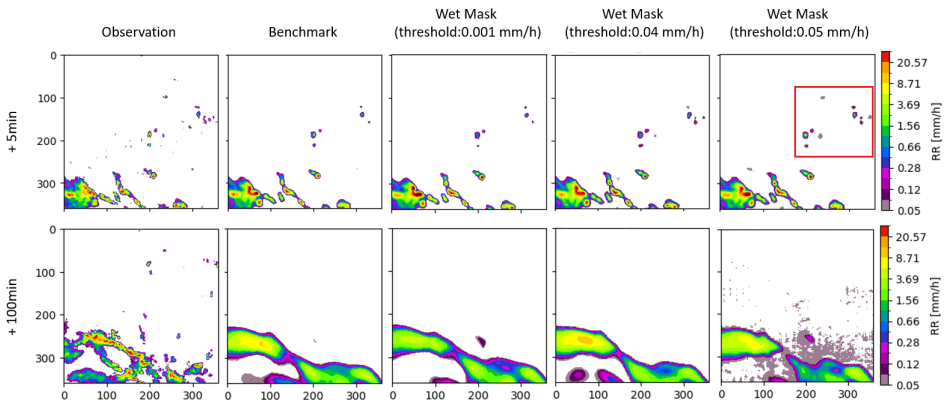

Figure 4.4: The prediction results with the application of wet mask and different thresholds.

The fact that the wet mask did not produce the expected results indicates that too many zeros in the training data might not be the main reason for the blurriness and decay of precipitation in the longer lead times.

4.2.3. Consequences of neighborhood loss functions

Max pooling

Three experiments with kernel sizes of 2, 7 and 15 are presented below (Figure 4.5). To our surprise, combining max pooling into the loss calculation actually made predictions worse across all lead times. Visually, Figure 4.5 shows that applying max pooling results in underprediction of rainfall intensities over the entire image, which is the opposite of what was intended (i.e., preserving high rain rates). Moreover, the max pooling approach also caused curious checkerboard patterns inside the rainfall fields. This checkerboard pattern becomes more apparent with larger kernel sizes. Interestingly, Lagerquist and Ebert-Uphoff (2022) also found this checkerboard pattern in their predictions; they explained it by the "excessive freedom" of the model during training, which makes the neural network produce erroneous small-scale patterns without penalty.

Let's assume there is a hypothetical prediction with the size of 9 by 9. In this hypothetical prediction, only 9 pixels are filled with the value of 1 (Figure 4.6). Then the max pooling with the kernel size of 3 by 3 will be applied to this hypothetical image. Although most pixels in the hypothetical image are zero, the resulting feature map after max pooling process can be filled up with the value of 1. This is because the sliding kernel can always cover at least one valued pixel in the hypothetical prediction and extract it. The feature map in this hypothetical prediction is identical to that in the ground truth (Figure 4.6), resulting in a loss value of 0 which means a perfect fit. This kind of model behavior is consistent with our findings about checkerboard patterns in the wet region. The checkerboard pattern could get rid of being correspondingly penalized and be preserved in latter iterations.

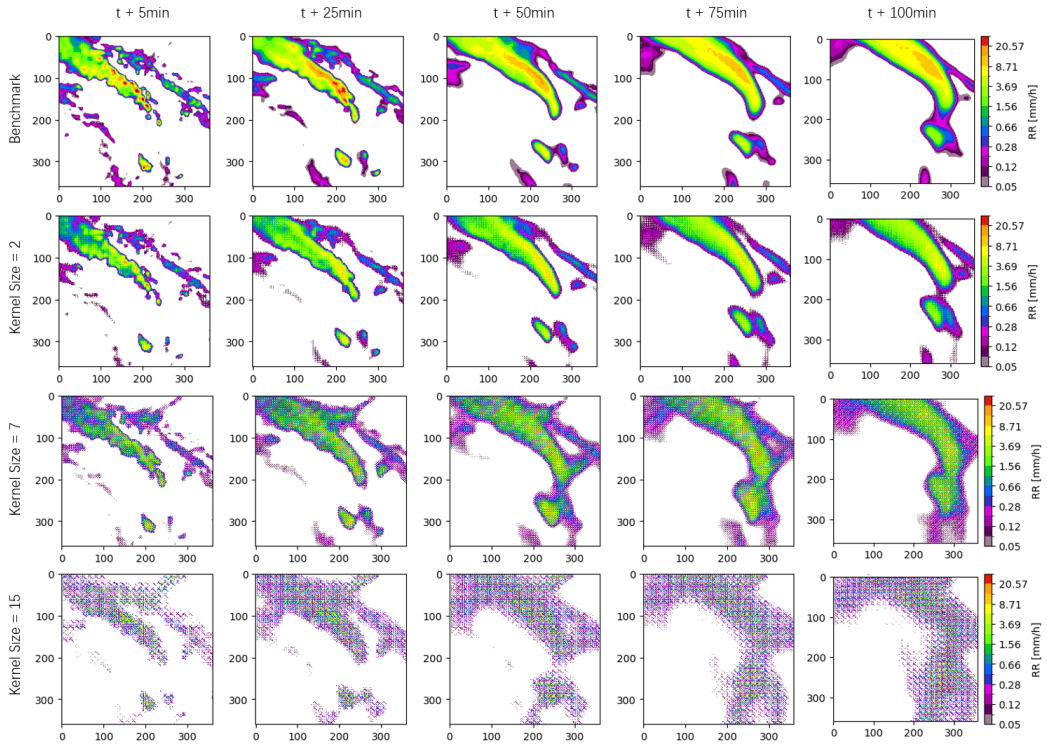

Figure 4.5: The predictions with the max pooling applied in loss calculation. Apart from the benchmark, it shows that the max pooling will cause a checkerboard pattern.

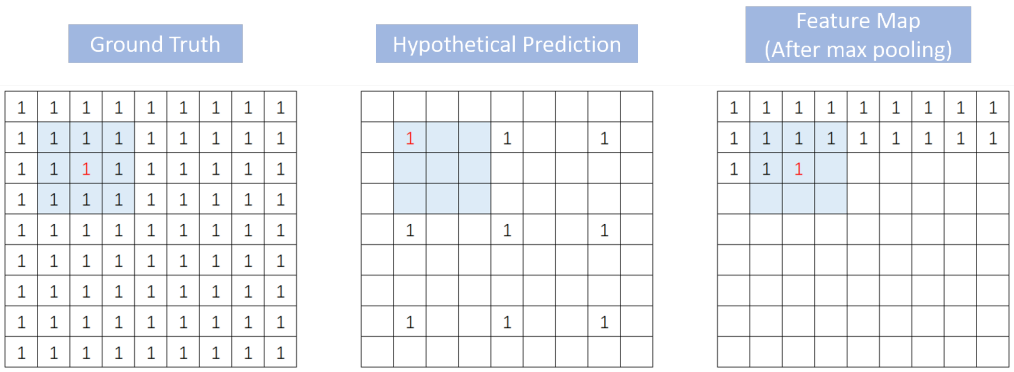

Figure 4.6: The illustration of a possible reason for checkerboard pattern; The maximum value within each sliding kernel is selected and becomes the output value for that corresponding central pixel in the feature map; The manually hypothetical prediction can store substantial zeros without penalty because the max pooling operation can ignore predictive performance in most pixels.

To avoid this checkerboard pattern, we tried to add a penalty to counteract the adverse effect of max pooling operation. For example, the average or min pooling can be combined with max pooling before loss calculation to prevent models from only predicting a few pixels surrounded by zeros. The resulting predictions are shown in Figure 4.7. The use of a min pooling mitigates the checkerboard patterns but does not completely remove them. The combination with average pooling can eliminate the checkerboard pattern, but this prediction closely resembles the benchmark's output. Also, the blurriness issue at longer lead times still persists.

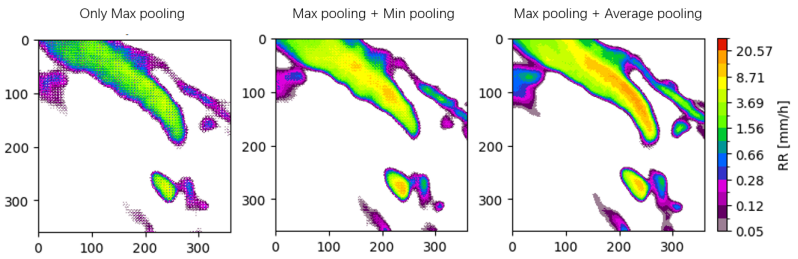

Figure 4.7: The effect of combining additional penalty with the max pooling. (Kernel size: 3)

Checkerboard patterns also emerge in the early stage of other trainings. Figure 4.8 displays predictions trained with the benchmark setting at 50, 90, 110, 150, 190, and 210 iterations. At the 50th iterations, there is distinct blocky pattern; however, this regular pattern gradually improves, and each pixel is filled with a value. When compared with neighborhood loss function, the pixel-wise loss function can help prevent it from occurring, but it does not fully resolve the issue. By the 210th iteration, the checkerboard pattern weakens, but we can still observe values of the same level appearing at regular intervals. This checkerboard pattern seems to be a natural phenomenon during the early training, and is not triggered by neighborhood loss. As it is not exclusive to experiments using the neighborhood loss, further research is needed to understand its underlying causes in the future.

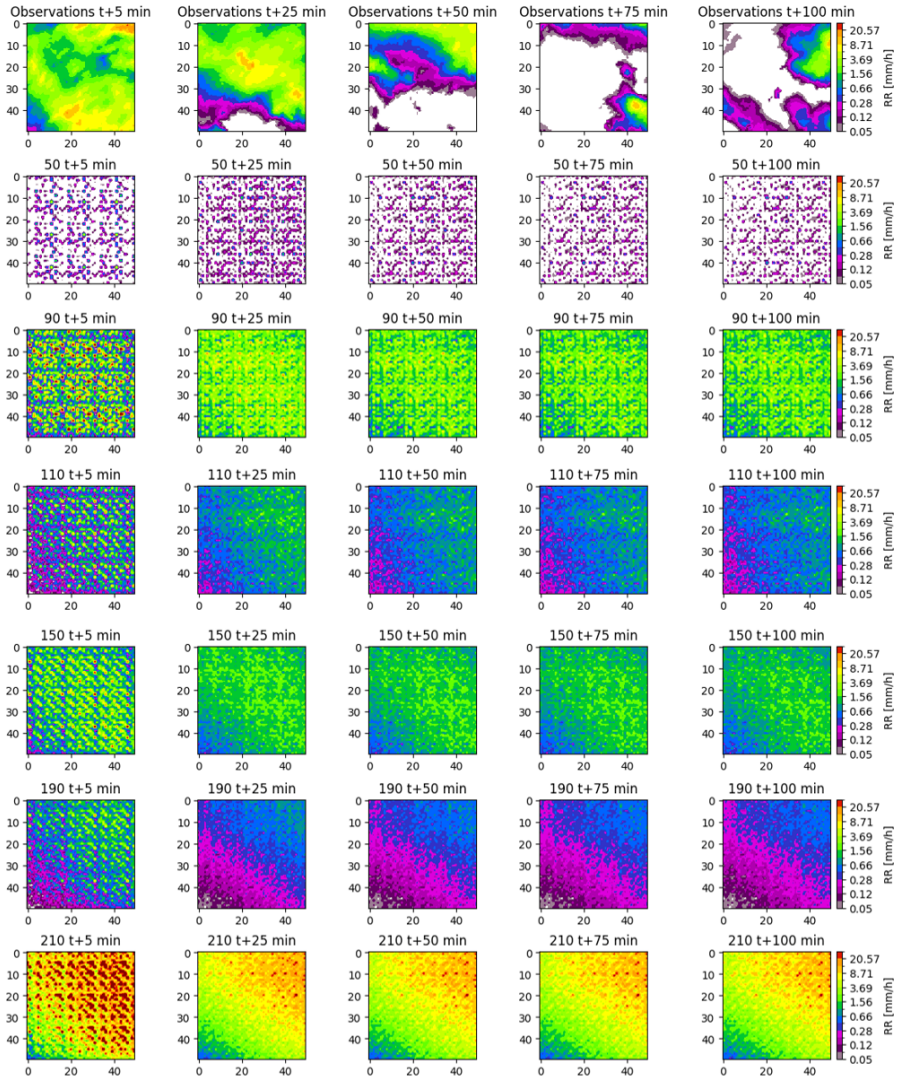

Figure 4.8: Without max pooling, checkerboard patterns are also found out at the early training stage of the benchmark.

Observation-oriented and Prediction-oriented loss

Fig 4.9 shows results created by the other way which interacts with neighborhood pixels in the evaluation. Checkerboard patterns also showed up during the observation-oriented evaluation. This time, the entire image including the dry area starts to contain checkerboard pattern. The reason is models succeed in finding another simple strategy to get satisfying loss results. By this simple strategy models only need to predict and adjust part of pixels, with the rest of pixels remaining zeros. This can be demonstrated by magnifying the prediction at early iteration. In Figure 4.10, a regular and organized pattern is created, in which the arrangement of colored pixels changes every iteration, in addition, the intensity level for each colored pixels is updated together. Turning to the prediction-oriented evaluation (Fig 4.9), related predictions are similar with the benchmark, without any significant difference. Similar with the max pooling, increasing the kernel size results in the underprediction of rain intensity.

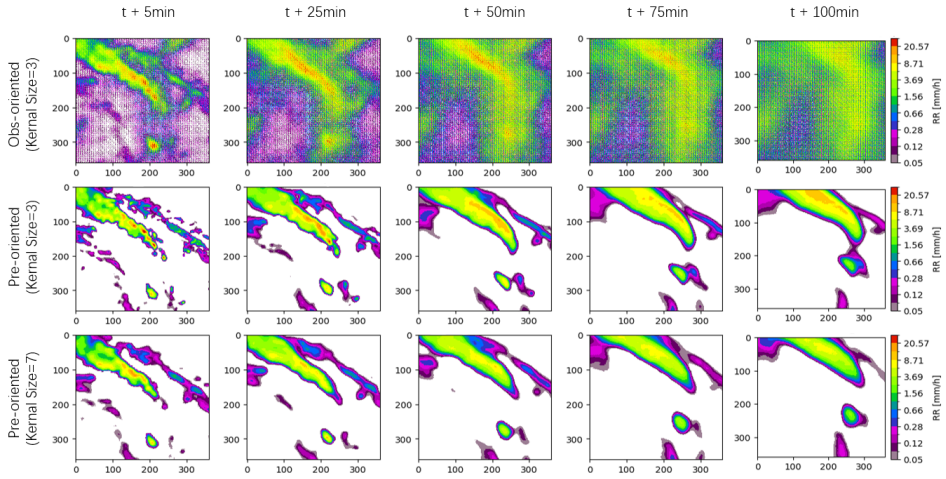

Figure 4.9: Prediction results with prediction-oriented and observation-oriented losses.

One of the main reasons for using max pooling during the loss calculation was to reduce the double penalty issue and allow the model to take more risks and predict higher rainfall intensities, even if they are not exactly at the right location. However, our results suggest that there is little usefulness in using neighborhood loss functions. This suggests that the double penalty is not the biggest issue when training a precipitation nowcasting model. This findings agree with the conclusion in Lagerquist and Ebert-Uphoff (2022).

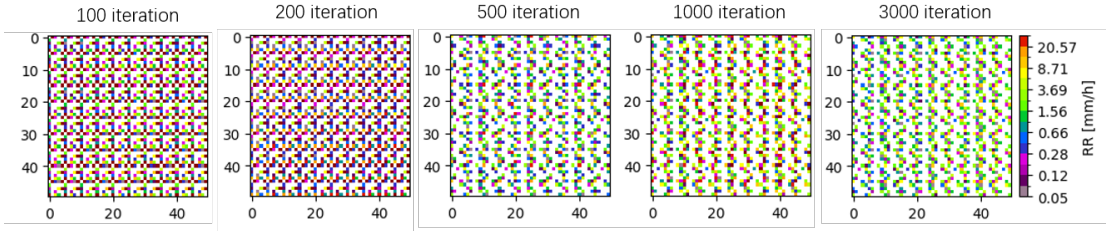

Figure 4.10: The magnified version of observation-oriented results at early iterations (t +50min); it illustrates how the model does the optimization. It showcases the mode change the blocky design and pixel intensity but keep retaining a significant number of zero values.

4.3. The effect of hyperparameter tunning

4.3.1. Batch size and learning rate

Figure 4.11 illustrates the visual differences associated with various batch sizes and learning rates. Regarding the impact of batch size, the figure demonstrates that increasing the batch size is able to significantly change the shape of rain fields. For instance, in the last lead time, models trained with a batch size of 8 manage to predict a sharply defined outline of the wet region. In contrast, the benchmark's predictions result in a more rounded boundary for the wet region.

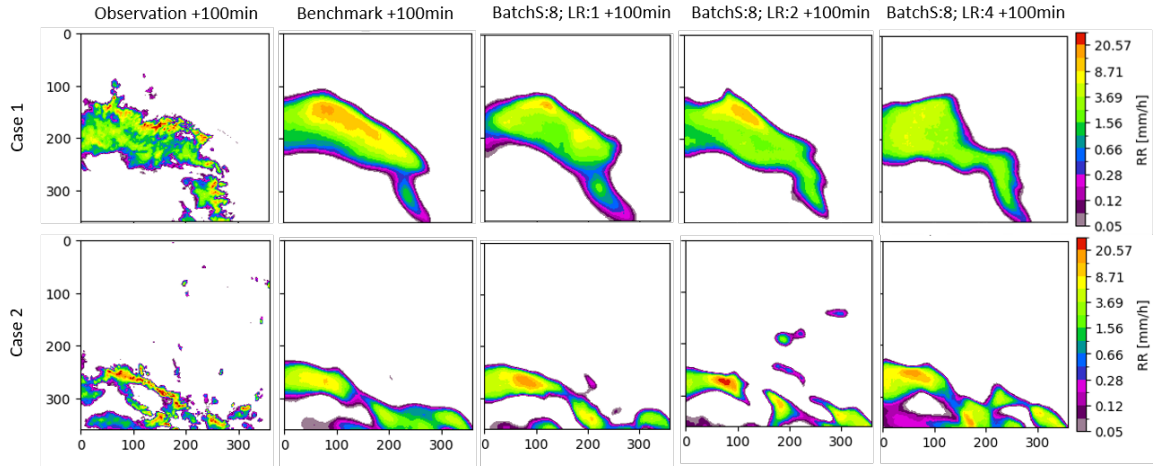

Figure 4.11: Predictions about experiments with batch size of 8 and varied larning rates in two events (t +100min).

Smaller batch sizes require smaller learning rates, while larger batch sizes allow for more substantial steps (Balles et al., 2017). Hence, the exploration of different learning rates is also conducted. Specifically, with a batch size of 8, learning rates of 1, 2, 4, 5, and 10 were investigated. The results show that a learning rate below 5 works well (Figure 4.11). When a larger learning rate is employed, the model's stability starts to be compromised. This is evident from the loss graph, where beyond a certain number of iterations, the loss value starts to fluctuate wildly and even reaches irrational levels, indicating the model's inability to effectively learn from the data.

From the loss graphs, whether a small or large batch size is used, the loss values reach a plateau at approximately 30,000 iteration. This suggests that the model's learning progress slows down significantly after 30,000 iteration. There are 185,324 sequences in our training set. At 30,000 iteration, Models with a batch size of 2 have got trained with 60,000 sequences, while models with a batch size of 8 have already utilized all available sequences. The metric results (Appendix A) show that the benchmark outperform others in most metrics, only except for MAE and GDL. This is consistent with other studys stating the larger batch size will lead to a degradation in the quality of the model (Keskar et al., 2017; De et al., 2017). But based on the visual analysis, a larger batch size is able to generate sharp and distinct boundary of rainy area. This shows the potential of reducing the blurriness regarding the rain field shape.

4.3.2. Number of filters

As decribed in the Methodology, the input images undergo three downsampling stages in the encoder. The precipitation information contained in radar images is transformed through convolutional operations into new images referred to as feature maps. The number of output feature maps depends on the number of filters employed in the convolutional operations. In this section, resulting predictions with decreasing amount of filters are presented in the Figure 4.12

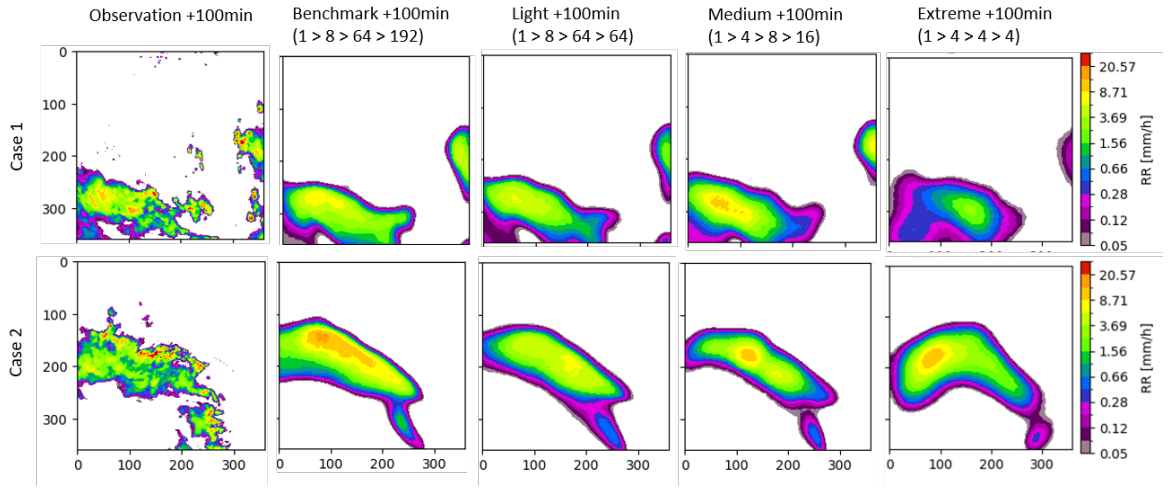

Figure 4.12: Forecasts about experiments with decreased filter numbers at the last lead time for two cases.

First, the predicted locations of the main rain field in three different downsizing modes align with the benchmark predictions (Figure 4.12). Even when reducing the number of filters from 192 to 4 in the top layer, the model consistently maintains the proper rain field location over time (Figure 4.13). However, regarding spatial details, it's evident that predictions become heavily blurred with fewer filters. While rainfall centers with relatively high rain rates are still distinct in Figure 4.12, they tend to be more centrally located within the wet region, showing reduced spatial bias. In contrast to the impact of a larger batch size, reducing the number of filters results in wet region outlines becoming more rounded and distorted at longer lead times. And in terms of intensity, fewer filters lead to an overall underestimation of the wet region, particularly along the boundaries.

TrajGRU improve the prediction by adjusting and updating the parameters in filters. Different feature maps emphasize a specific aspect of spatial characteristic. For example, some feature maps will stress the edges or gradient in the left-hand side of wet regions. In the image recognition literature, an increasing the number of filters is needed to ensure that information is not lost too much when down-scaling (Tran and Song, 2019). However, in the prediction task such as this case-study, using too many filters could make the models suffer from the over-fitting problem. Because at a higher abstraction layer, a model is expected to make assumption about the global rather than the local variations (Tran and Song, 2019). In our case, local details within the wet region significantly decide the sharpness of prediction. As mentioned in section 4.1, the benchmark shows its weakness of the prediction at local scale (Figure 4.1). As the image undergoes downsizing in the top layer, this valuable local information converted and included in the feature maps, becomes diluted. At the same time, excessive feature maps are generated from images with limited local details, displaying slight difference in global features. With too much redundant global features, models may fail to predict effectively. Based on our visual results (Figure 4.13), after decreasing the filter number at the last layer from 192 to 64, there is no significant changes in the inner spatial structure, implying the two-thirds of filters at a highest layer don't significantly contribute to prediction. Thus, there is no need to make 192 kinds of assumptions at global scales as it will generate many similar feature maps.

Additionally, there's a significant reduction in model size, with the trained model's file size dropping from 48.7 MB in the Benchmark to 9.35 MB in 'Light' mode Table 4.1. For the 'Extreme' mode, the model size can shrink to smaller than 1 MB. In general, smaller models can be applied and updated by the platform more rapidly.

Table 4.1: Total parameter number and model size after decreasing filter number.

	Benchmark	**Light**	**Medium**	**Extreme**
Total parameter number	12,753,909	2,444,917	323,913	159,377
Model size	48.7 Mb	9.35 Mb	1.26 Mb	648 Kb

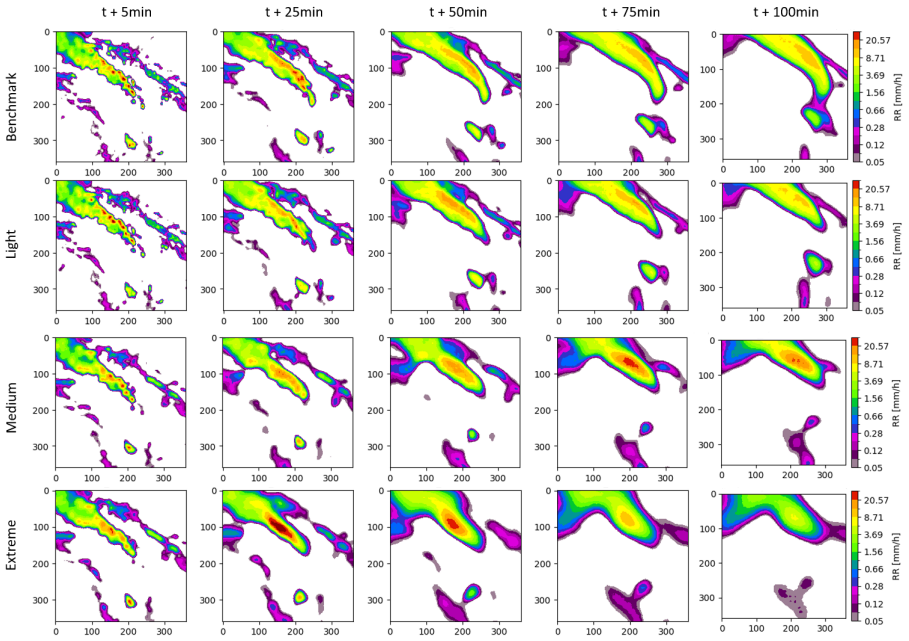

Figure 4.13: Predictions of three modes in terms of decreased filter number with different lead times.

According to the metric results (Figure 4.14), we observed that the "Light" mode achieved scores similar to the benchmark, with better performance at 4, 5, and 6 frames. On the other hand, the "Medium" and "Extreme" modes, on average, performed worse than the benchmark. However, the "Extreme" mode managed to achieve scores comparable to the benchmark, particularly in the last few lead times, except for the balanced loss which we used as the objective function. In fact, the "Extreme" mode exhibited a smaller MAE, GDL, and a larger SSIM than the benchmark, indicating preferable performance. Initially, there was concern that the "Extreme" model might lead to the breakdown of TrajGRU. However, both metrics and visual analysis demonstrate that even with extremely fewer filters, the model still performs logically.

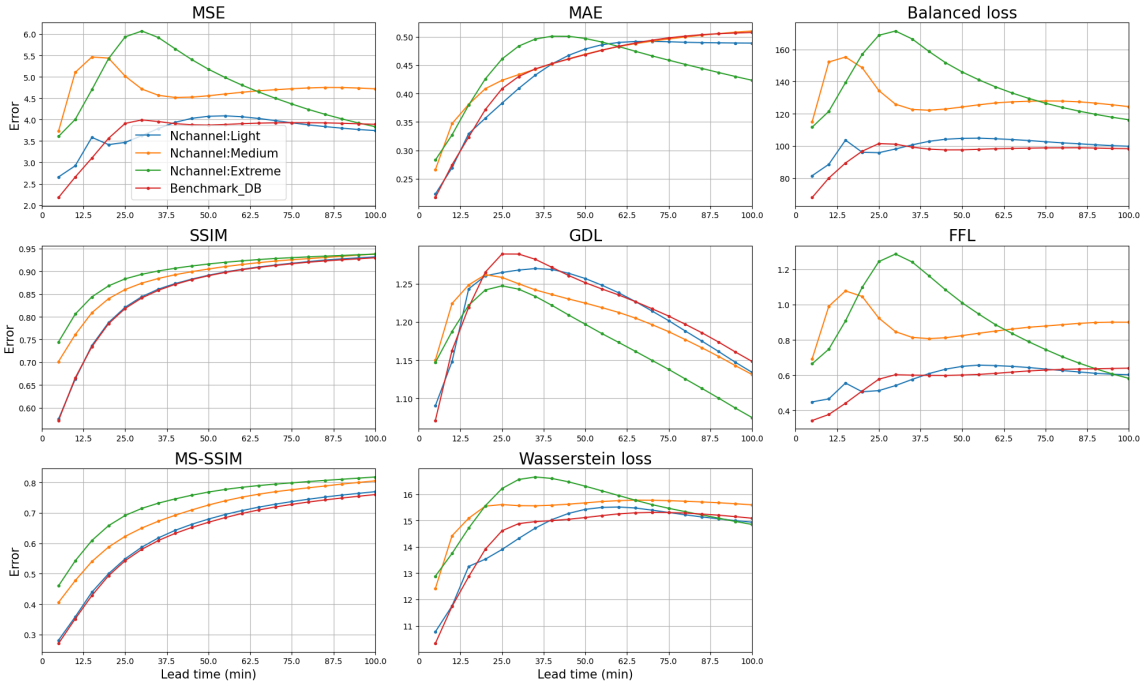

Figure 4.14: Metric results for experiments with decreased number of filters.

The pruning technique is to reduce the extent of a neural network by removing unwanted and insignificant parts. In this thesis, we focus on the filter pruning and randomly reduce filter number in the top layer. Many studies related to the weight pruning demonstrated that eliminating unnecessary weights from neural networks (Han et al., 2015; Li et al., 2017; Frankle and Carbin, 2019) can reduce parameters by more than 90% without harming accuracy. Frankle and Carbin (2019) innovated an iterative pruning strategy, cutting down some connections with least importance in multiple rounds. They found that smaller networks are able to retain the same level of accuracy and can even go higher. As we have observed that randomly removing filters in our current network can maintain similar performance, future studies can explore different pruning methods and further compress the model network to identify the most optimal and efficient mode.

4.3.3. Multiple parameter groups

In the Benchmark scenario, isolated showers with high precipitation rates remain fixed within the wet region. There's minimal variation in the spatial structure of the rain field over time, only with finer details getting increasingly blurred. In this section, the performance of multiple parameter groups will be displayed. The introduction of multiple parameter groups aims to provide the model with more freedom and flexibility. In Figure 4.15, We have extracted 6 frames from the middle position (+30 to +75 minutes) of the entire sequence, as special changes are concentrated on this interval.

For the benchmark in the first row of Figure 4.15, those fine details within the rain field at the first lead time gradually blur and merge together with increasing lead times. The orange patch representing the high intensity keep shrinking with increasing lead times. Besides that, there is no significant change observed within 45 minutes. When employing 4 sets of parameter groups, some isolated showers will start to emerge and dissipate among these 6 frames, referring to patches that are circled in Figure 4.15. Although there is only one or two such smoothed red patches popping up in the sequence, this dynamic changes will not occur in the benchmark or other experiments. Additionally, expansion of the high rainfall in the wet region area can also be found out in the last 25 minutes. Thus, more spatial dynamics start to show up with the involvement of multiple parameter groups. Compared to a consistent fading of rain, the dynamic pattern aligns more closely with reality.

The trained model, which employs uniform learning rates set at 1 for all four parameter sets, exhibits

a fading-out pattern similar to the benchmark. In this configuration, there are no sudden high-intensity appearances or expansions in the rain field after the initial lead time. Conversely, assigning higher learning rates to later lead times results in certain spatial dynamics and variability (refer to Figure 4.15), effectively manage to break the continuity of predictions. These momentary instances of rain appearance can represent the growth and evolution of rain. Hence, there is a need to set different learning rates to different lead times.

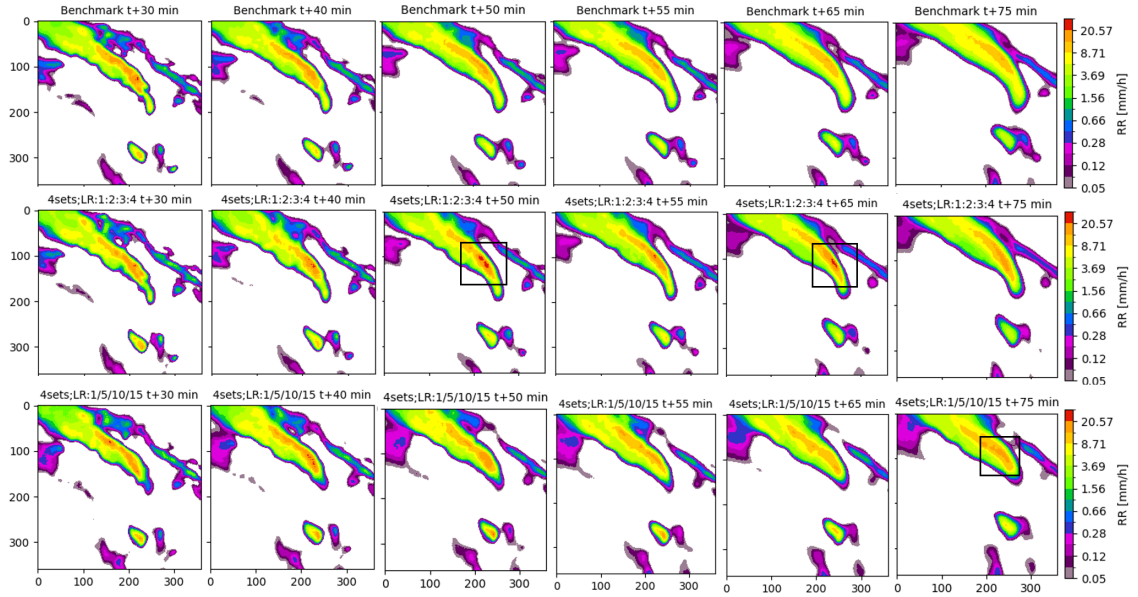

Figure 4.15: The prediction results using 4 sets of parameters and learning rates.

In this approach, the metrics result exhibit irregular curves with increasing lead times when compared to other trials. Unlike the consistent and smooth trend observed in the metric curves of other experiments, using four different sets of parameter groups results in a fluctuating loss curve (Figure 4.16. This fluctuation in the loss curve aligns with those visual dynamics observed and found out in this approach. For the first few lead times, the benchmark always performs the best, while experiments with large learning rates yield abnormally high values in MSE, Balanced loss, and FFL. However, upon visual analysis, they are able to produce similar rain forecasts. Both SSIM and MS-SSIM did not be displayed here, as the loss curves of all experiments closely overlap.

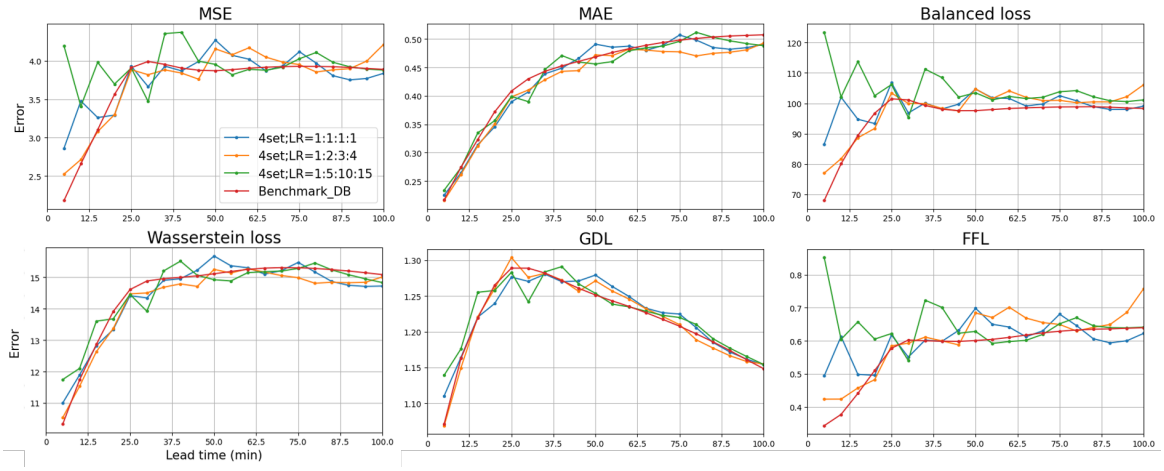

Figure 4.16: Metric results with 4 parameter groups.

4.4. The effect of changed model inputs and outputs

4.4.1. Time-differenced rain intensity

In previous experiments, radar image sequences have been used as both model input and output. However, we are now exploring the utilization of different types of data, such as the time difference of precipitation, as input and output for the model. The outcomes of two experiments are presented in Figure 4.17.

It is interesting to find that the resulting forecasts exhibit intricate structures within the wet region, not only during the initial lead time but also in the final lead time (Figure 4.17). The model didn't blur out the spatial structure; and the isolated showers with high precipitation rates show up in different location with increasing lead times. Furthermore, the choice of weighting strategy influences the prediction features. Images trained with the original weighting strategy depict more extreme precipitation inside the rain field. This divergence stems from the distinct prioritization of stressed pixels. The original strategy emphasizes pixels with high rain rates, often positioned within the middle or interior of the wet region. Conversely, assigning pixel weights based on the absolute time difference values emphasizes pixels along the edge of the wet region (Figure 4.19). Contrary to the visual result, which displays impressive sharpness, the metric analysis does not agree with such improvement (Appendix. A), but rather indicates a completely inaccurate prediction.

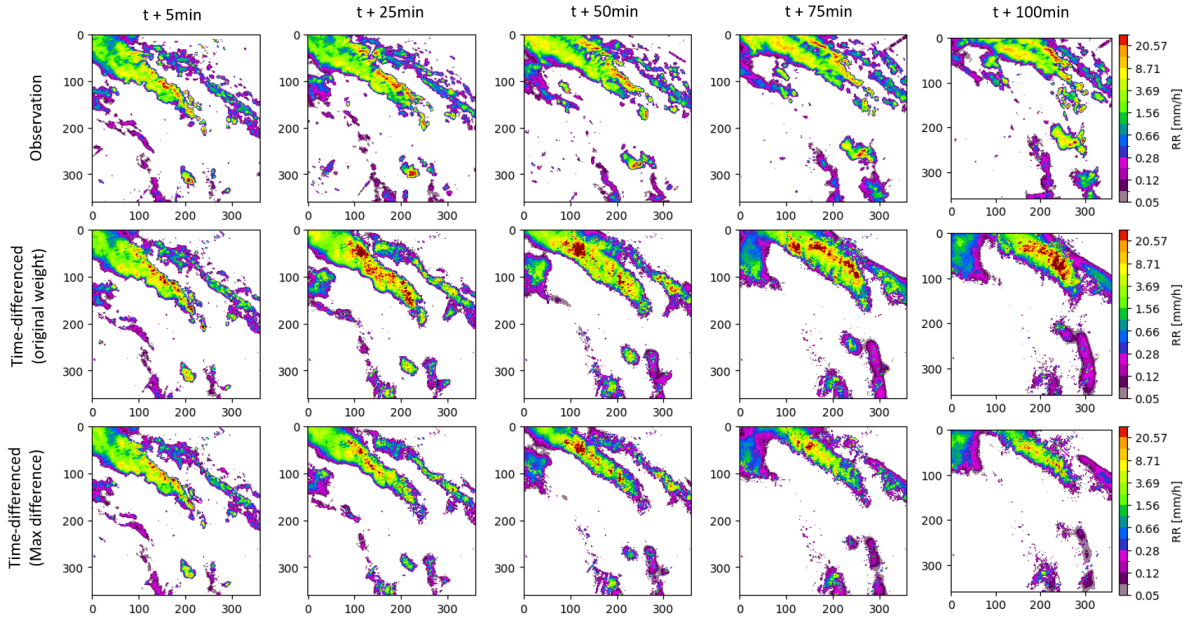

Figure 4.17: Predictions by using time-differenced images as model input and output. Two weighting strategies are studied, in which original weights represent Equation 2.4.

However, in comparison to observations, these isolated showers and fine structures are not positioned accurately. It remains challenging to comprehend how the model predicts these intricate patterns. The underlying logic guiding the model's determination of isolated shower locations remains unclear. To enhance our understanding of precipitation evolution over time in the output sequence, we have incorporated GIFs as an additional analytical tool. Through this approach, we find that these fine structures (regions of extreme precipitation) remain fixed in image position as time progresses. Isolated showers emerge when covered by rainy areas and abruptly vanish when the rain shifts away. Their positions do not alter with the movement of rainfall zones. The finding of isolated shower evolution in GIFs prove that corresponding predictions are unrealistic and contrived.

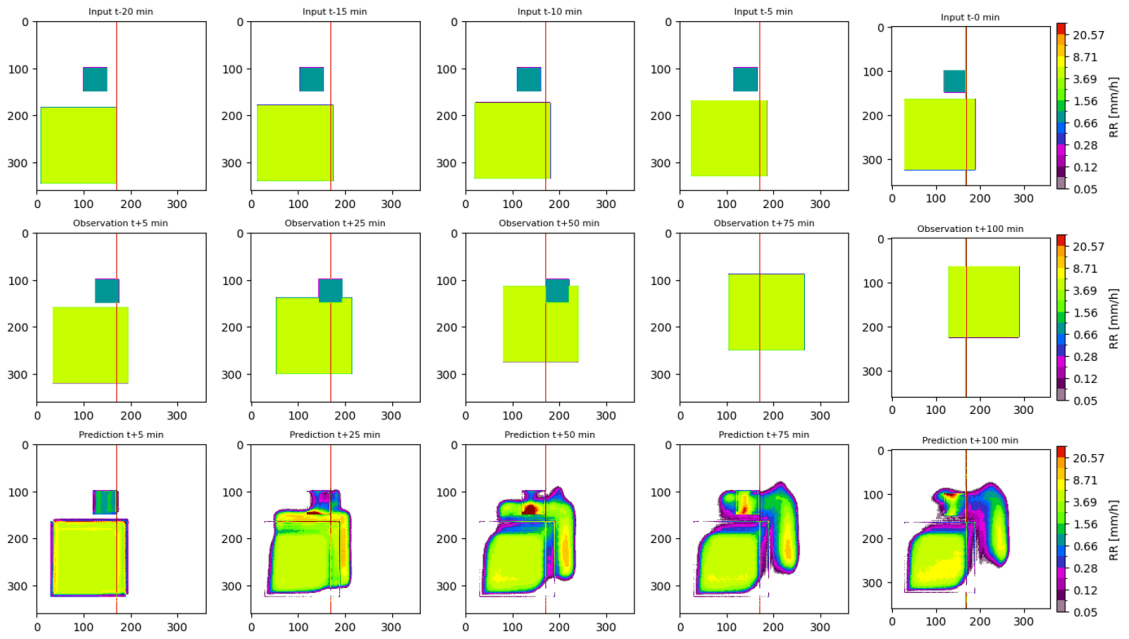

Figure 4.18: The illustration of the ghost formation when using time-differenced images. Models struggle to predict correct time-differenced rain intensity to remove the ghost from the last input (t + 0min).

The origin of these unrealistic fine structures can be attributed to the radar image at 0 minutes. As both the input and output of the model are time-differenced NPV images, the generation of predicted radar images involves adding time-differenced images to the radar image at 0 minutes (Figure 3.5). An artificial example in the Figure might help to explain this phenomenon. In this scenario, the main wet region moves from the lower left to the upper right, accompanied by a small independent rain field at the upper part of the main wet region, moving at a steady pace from left to right (Figure 4.18). Actual observations reveal that the two independent rain fields move separately and converge at +25 minutes. However, in the model's predicted frames, it fails to anticipate the movement patterns of both rain fields (Figure 4.18). The model provides relatively more accurate predictions for the larger moving rain field compared to the smaller one. Additionally, the position of the upper small rain field in forecasts remains stationary and identical over time. When the two rain fields overlap, the rainfall intensity of the overlapping section amplifies, similar to the phenomenon seen in the prediction of the realistic case (Figure 4.19). Consequently, it is possible that the irrational isolated showers depicted in the Figure originate from the overlap of the two rain fields.

The predictions of rainfall time differences are presented below (Figure 4.17). Although the models appear to produce sharp forecasts of radar images, they still yield blurred predictions of time difference values over time (Figure 4.19). Time difference of rainfall represents the direct model output. The ultimate predicted radar images are formed by combining the radar image of the current time (+0 minutes) with the cumulative predicted time difference. Based on the above analysis, the sharpness of the final prediction likely originates from the details in the present time's radar image. However, due to the limited predictive capacity of models for time difference values, these details cannot be promptly eliminated or counterbalanced. As a result, they persist in the same positions across varying lead times. Overall, fine rainfall structures shown in predictions (Figure 4.17) are not predicted by trained models but the ghost of radar input.

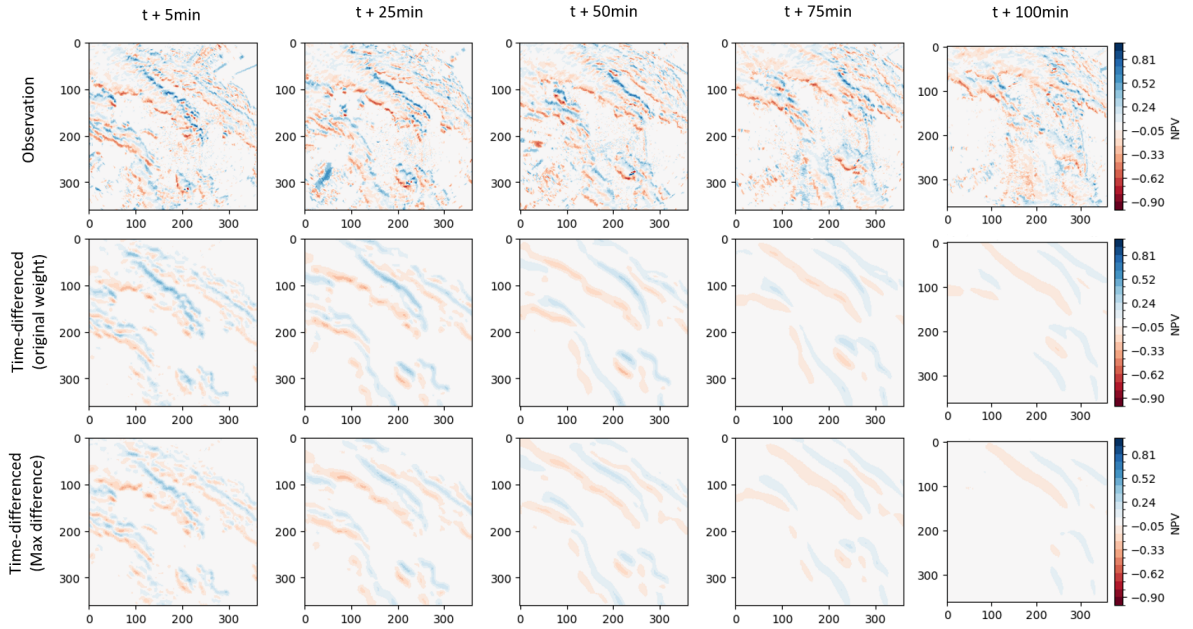

Figure 4.19: The direct time-differenced outputs before adding with the radar image of the current moment (t +0min).

4.4.2. Multiple model inputs

In the following section, we will present the model's performance in integrating NPV images with other types of model inputs. Unlike rain fields' shape got changed through the use of larger batch sizes or more internal dynamics predicted by involving multiple parameter groups, using multiple model inputs shows that this approach is able to refine the rain fields with more details. The visual predictions (Figure 4.22) with inputs combining time-differenced maps, gradients maps in the y direction, and maps only storing pixel values larger than 0.854 NPV, show that the blurriness and fading out did not get solved. Among them, the experiment incorporating RR images as the additional model input is the most promising. This approach enriches the wet region with more internal details (Figure 4.20), especially at the first lead time. The Figure 4.20 magnifies a specific part of the rain field, which displays the actual intensity of the individual pixel. It shows an enhanced performance of the internal rain field with finer structures in comparison to the Benchmark. In the benchmark, the pixels with the same value are more closely packed together in certain areas, resulting in a concentrated distribution pattern (Figure 4.20). When incorporating RR images as the additional model input, the high-intensity pixels can become more scattered and detailed. The overall blurriness is reduced to some extent, while some details may not precisely match the reality.

The fine details can be derived from the higher contrast between the highest and the lowest rain intensities inside the rain field when perceiving radar images on the RR domain. This is previously mentioned in Dekker (2022). Figure 4.21 displays radar observations in different domains. Both images do a logarithmic normalization in the colorbar. The upper and lower limits of the colorbar are defined according to the maximum and minimum values within their respective domains. Observations have a maximum value of 1 in terms of NPV and a maximum of 48.6 mm/h in RR. The contrast of values inside the rain field emphasizes fine structures and contributes to its prediction. In the interior of the rain field, Figure 4.21 shows that the NPV image has low contrast with a smaller range of pixel values falling between 0.7 and 1 within the precipitation field. With such subtle contrast, models can not detect spatial patterns. Radar images presented in the RR domain convey more informative details and gradients, which enable the model to learn effectively.

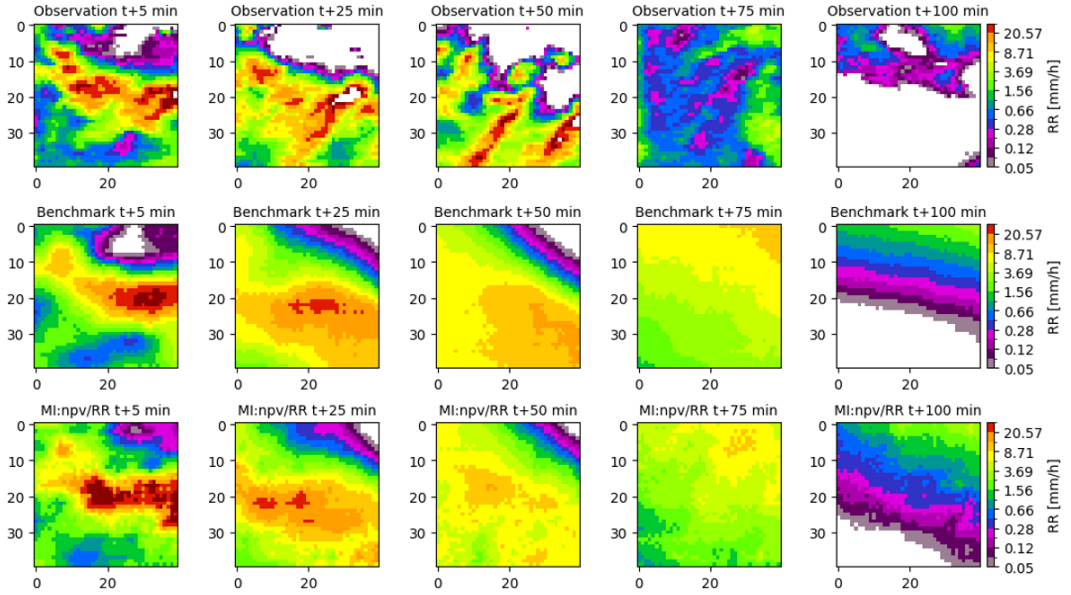

Figure 4.20: Regional details after magnifying a specific part of rain fields; the first row is the ground truth, the second row shows the benchmark, the third row illustrates the prediction combined radar images in RR as an additional input.

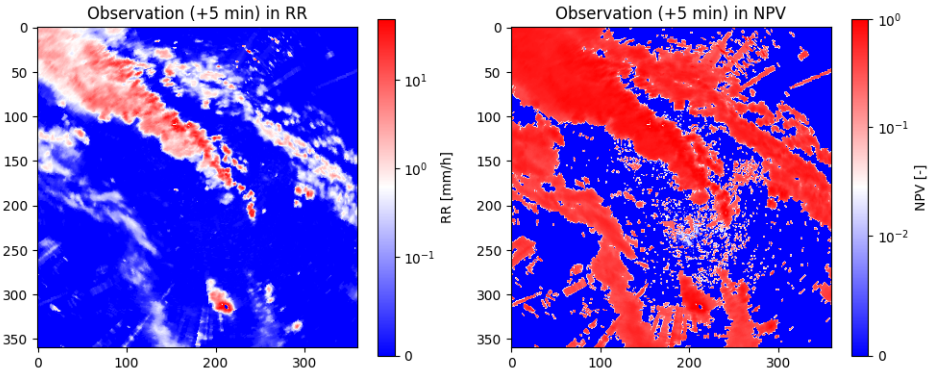

Figure 4.21: Radar images in RR and NPV domains exhibit opposite patterns within the rainy area. RR images show high contrasts within rain fields, indicating greater variation in pixel values. On the contrary, NPV images show a smaller range of pixel values within rain fields.

It should be noted that, except for the radar image in RR, all other transformed model inputs (gradient images or time-differenced images) are displayed in the NPV domain. After we understand that enhanced contrast can help the model extract features effectively, future work should consider RR values rather than NPV for all additional datasets. Because the underlying distribution of pixel values inside the precipitation field is more important than the overall distribution of the entire image. Using data in RR has the potential to amplify the gradient and sharpen precipitation fields.

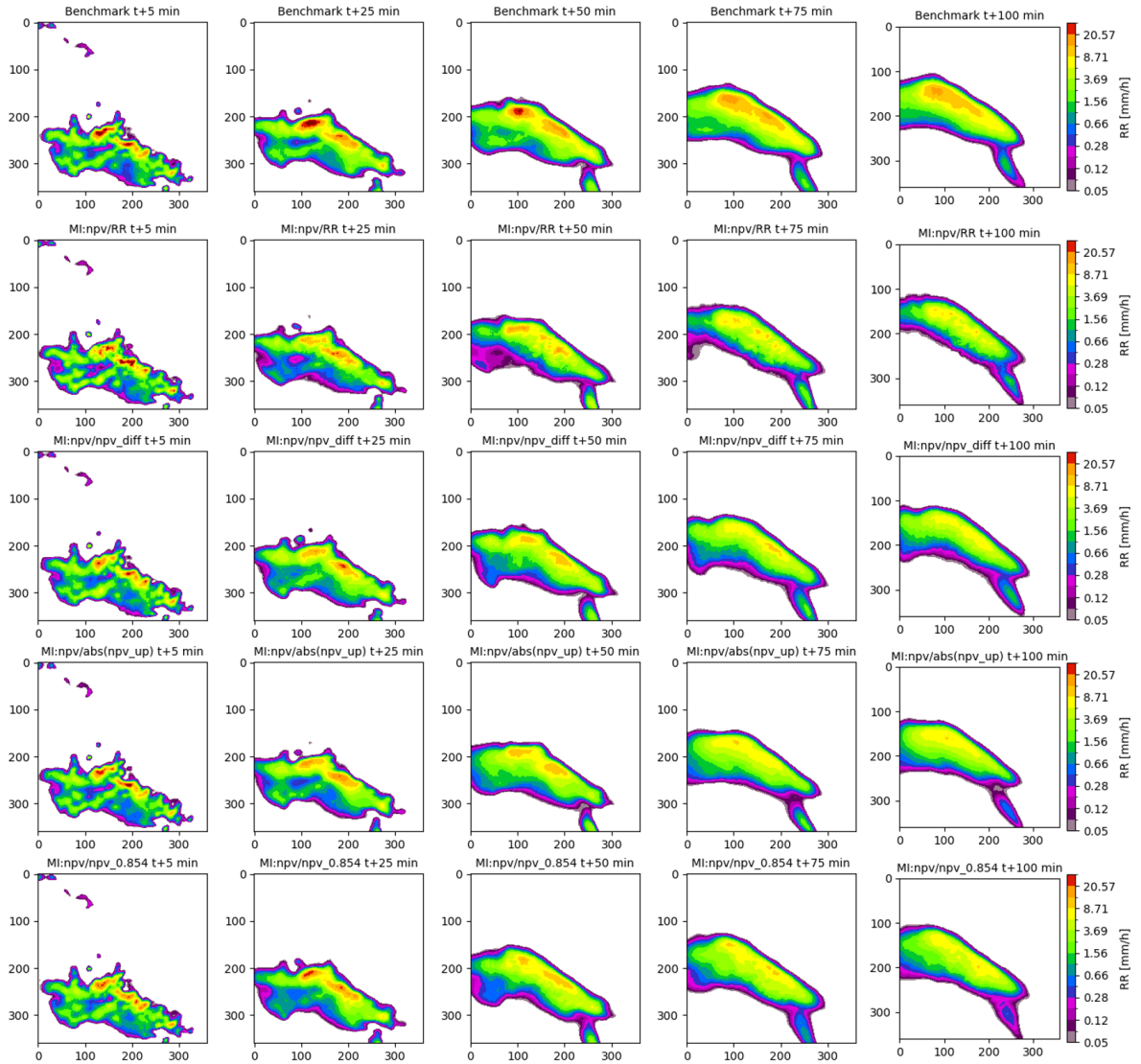

Figure 4.22: Results from different combinations with various model inputs.

Table 4.2: An overview of all results from above approaches. Approaches are divided into three main categories based on which component in TrajGRU we make changes.

Category	Approach	Highlight
Loss function	Temporal weight	• Temporal weights did not lead to any improvements in sharpness for all lead times;
	Wet mask	• Setting a higher threshold for wet pixels can sharpen the prediction at first lead time, but start to overpredict dry pixel values; • There is no any improvement in the whole; • Data imbalance is not the contributing factor for the blurriness;
	Max pooling loss	• The checkerboard pattern is found in predictions; • Max pooling operations help models store substantial zeros without penalty;
	Prediction-oriented loss	• Checkerboard pattern found in results trained by the observation-oriented loss; • No predictive improvement with the prediction-oriented loss; • The double penalty issue is not the crucial factor for the blurriness in the current model;
Model hyper-parameter	Batch size and learning rate	• With larger batch size and learning rate, the shape and boundary of rain fields get sharper and angular; • The learning rate is crucial; a large learning rate leads to bad model convergence;
	Number of filter	• The model performance did not degrade after the model size was reduced 5 times;
	Multiple parameter groups	• More rain dynamics show up in animation. High intensities can be found in latter lead times, instead of just the fading out; • But the blurriness and fading out persist in predictions;
Model inputs and outputs	Time-differenced dataset	• Predictions at all lead times display fine but unrealistic structures; • Those fine details are the ghost from the last input frame (t +0min);
	Additional model inputs	• The prediction trained with RR images gets sharper at the first lead time; • The blurriness and the fading out still persist in predictions;

5 Conclusion

This thesis began with the goal to address the remaining problems from Dekker (2022): data imbalance and double penalty, which were believed to be the two major causes responsible for the smoothing and fading out of the forecasts. To achieve this, various changes were made to TrajGRU, and they can be generally be grouped into three parts: 1) the use of new weighting strategies, 2) the use of neighborhood loss functions, and 3) modifications to the model structure of TrajGRU. This final chapter summarizes our findings and answers the research questions.

5.1. Answers to research questions

SQ1: Can model performance be improved by applying temporal or spatial weighting strategies aimed at addressing data imbalance?

Our results show that models in which the error weights decrease with lead time may slow down the decay of high intensities. However, the alternative weighting strategies did not solve the blurriness issue and fading out of rainfall at longer lead times. The wet masks that forced TrajGRU to exclusively focus on predictive performance within precipitation regions, and was supposed to mitigate issues related to data imbalance between dry and wet pixels, did not substantially improve the results either. The resulting predictions were similar to the benchmark, and their blurriness increased with lead time. One advantage of the wet mask approach when combined with a larger threshold, is that it can potentially sharpen a bit the predictions at short lead times compared to the benchmark. However, overall, the weighting and masking strategies did not significantly contribute to improving the sharpness of the predictions.

SQ2: Is it possible to sharpen the predictions and avoid the double penalty issue by using spatial neighborhood loss functions?

Our results show that models trained with such neighborhood loss functions exhibit strange "checkerboard patterns", in which only a small fraction of pixels have a positive rainfall value, and the others around them are zero. Lagerquist and Ebert-Uphoff (2022) also mentioned this pattern in their work, and attributed it to and excessive freedom issue during parameter adjustments (i.e., the fact that many weights and biases can be changed simultaneously without affecting the loss). However, the excessive freedom issue does not explain why good models that have realistic spatial structures (e.g., the benchmark) systematically converge to checkerboard patterns when trained with neighborhood loss functions. This strange behavior holds regardless of the starting point, learning rate and optimizer used during the training. The emergence of checkerboard patterns and how to avoid them is something that should be investigated in more detail in future work. It is crucial for determining the role played by the double penalty issue in controlling the blurriness of precipitation forecasts.

SQ3: Are there specific elements within TrajGRU, such as hyperparameters and model inputs/outputs, that can be adjusted to enhance predictive performance?

Our results show that larger batch sizes lead to better models with sharper edges and better defined rainfall cells. However, the larger batch sizes did not reduce the overall blurriness of the predictions at longer lead times. The shrinking experiments during which TrajGRU was retrained with a smaller number of spatial filters showed that the benchmark contains a lot of unnecessary complexity. Smaller models that use fewer filters in the convolutional layers were able to reproduce very similar structures. This finding implies that some parameters in the benchmark are redundant and can be removed to enhance the robustness and speed of the model while reducing its size.

Another improvement to TrajGRU that way explored in this thesis consisted in dividing the shared parameter set in the forecaster into four distinct groups. One of the limitations of the benchmark is that it uses the same parameter set to forecasts all 20 lead times. To give the model more flexibility, we performed experiments in which the parameter sets were divided into 4 different groups, one group for every 5 frames. Our results show that with this approach, rainfall fields do not systematically fade out with increasing lead time anymore. Instead, small-scale isolated showers can emerge and intensify at intermediate lead times. We conclude that the use of different parameter sets for short, medium and extended lead times appears to be a rather promising approach for combating the fading out of rainfall fields. However, it also increases the size of the models and adds new parameters, which need to be properly adjusted. The choice of the learning rate plays a key role in this approach and we recommended to assign larger learning rates to longer lead times.

Finally, the influence of changing the model input was tested. In the initial phase, we replaced the standard radar images with time-differenced images. This approach yielded sharper predictions for all lead times. However, it also resulted in unrealistic space-time patterns where high intensity rain cells stayed at the same location over multiple lead times. We think that these stationary structures (i.e., the "ghosts" of previous radar images) emerge because of the natural tendency of the model to predict zero changes in rainfall intensity from one time step to another. To avoid these ghosts, we combined the time difference inputs with the original radar images. However, the resulting predictions looked very similar to the benchmark and were not sharp anymore. Only the experiments which used the rainfall rate as an input instead of the normalized pixel values exhibited some slightly finer structures.

Research Question: How can TrajGRU be upgraded to mitigate issues related to the prediction blurriness and fading out?

Overall, the blurriness and fading out issues still remain open challenges that need to be investigated in more detail. The loss function modifications, which in theory, have the potential to alleviate data imbalances and avoid the double penalty issues, did not solve the blurriness issue. In some cases, like for the experiments with the neighborhood loss functions, it can even lead to other undesirable effects such as the appearance of a checkerboard patterns, which requires further investigation. Based on these results, we conclude that the loss function does not appear to be the most important factor limiting TrajGRU's performance, and that there are likely other, more important, underlying factors that are responsible for the blurriness and fading out.

By contrast, modifying the model itself and its architecture seems more promising. A larger batch size, for example, reduces the blurriness of rain field boundaries at longer lead times, and adding more parameter groups helps mitigate the fading out of high-intensity predictions. Changing model inputs also has the potential to enrich precipitation fields with more detailed spatial patterns. However, this is not guaranteed and can also lead to unrealistic space-time patterns, as demonstrated by the experiments conducted on rainfall difference maps. It is important to note that in this thesis, changes were mostly assessed through visual analysis of individual predictions and animations. But sharper images alone may not necessarily be better and a more comprehensive and quantitative analysis of each change is required before final conclusions can be drawn.

Although it already performs well, the current model still appears to have some limited room for improvement. In this thesis, a lot of ideas were explored, but there's still much more to investigate. The following section provides some recommendations for future work by highlighting the critical components that should be investigated in more depth.

5.2. Recommendations

Instead of tweaking the loss functions, we recommend shifting the attention to the model network structure itself. One recommendation is to continue to optimize the number of filters in different layers in TrajGRU. When exploring this approach, one should keep in mind that the feature maps corresponding to the top layer represent spatial features from a global perspective. After the convolution operation is applied, a feature map is generated for each filter. Each feature map highlights specific features or patterns present in the input image. These features can include edges, corners, textures, or more complex patterns, depending on the layer and filter. Compared to global-scale spatial features, the current model conveys insufficient local-scale spatial information, which may be the cause of the blurry predictions. Referring to Table 2.1, in the initial convolution process, the input image is transformed into only 8 feature maps, with the image size reduced to one-fifth of the original radar image. Possibility, a great deal of spatial information is lost during this downsampling process. In the top layer (Table 2.1), referred to as 'edown2', although the image size is halved, the number of feature maps remains the same, which also indicates possible information loss. To address this issue, I propose to reduce the redundant feature maps in the top layer and increase the number of filters in the bottom layer. Alternatively, one could enlarge the output size in the top layer, since the resizing of the image from 480x480 to 16x16 is a big reduction.

Regarding the number of parameter groups: in this thesis, the 20 frames corresponding to the 20 different lead times were split into 4 distinct parameter groups. However, one could also consider a larger number of groups, up to 20. Our results show that some (small) variations in rain dynamics are possible using this approach. But the potential has not been fully explored yet. When working with multiple parameter groups, we recommended to increase the learning rate as lead times progresses. However, finding the right balance can be hard, as excessively large learning rates might also cause the optimization process to jump over the optimum and the loss curve to exhibit large fluctuations. For example, one experiment we conducted with learning rates set to 1, 10, 20, and 30 did not train properly. The tuning of learning rates is something that should be looked at in more detail but can also be very time-consuming.

Another promising approach is to use multiple model inputs during training. For example, in addition to using NPV, we recommend to also incorporate radar images in RR into the model input. In other relevant experiments, such as combining the original radar image with time-differenced data, all of those additional input data are in NPV. Testing all images in the RR domain, rather than NPV, is recommended. Since RR exhibits more significant value differences between pixels, this feature should also be reflected and have an impact on time-differenced radar images or gradient images in the y-direction. Similarly, Dekker (2022) recommended applying alternative data transformations to rainfall rates, such as the log transformation. This idea is to balance the dataset and magnify the gradients within the rain field. On the other hand, the inclusion of other data types such as Echo Top Height (Elsmann, 2023), wind speed (Kaparakis and Mehrkanoon, 2023) or numerical weather prediction data (Niu et al., 2023) is also promising and worth trying. In this regard, various information in terms of temporal and spatial aspects can be blended into the predictions. One important thing to keep in mind is the value distribution: whether this specific data distribution amplifies value gradient within the rain field rather than at the edge of the rain fields. Alternatively, transforming data into a different unit, such as using RR, can enhance image detail.

As we evaluate the effectiveness of each approach in our results, you can observe that each approach has a distinct influence on the predictions; some introduce more internal dynamics, while others enhance the prediction of finer structures. The influence of each approach is currently subtle, possibly because the effect of each approach is constrained by the others. To continue forward after this research, it is important to move away from using only radar data and incorporate more understanding into machine learning models. I think the addition of other atmospheric information could help sharpen the predictions, but it may not fully address increasing blurriness over time, as this issue is related to the model architecture. Thus, we also need to further optimize the model structure to help TrajGRU adjust its predictions over time and space as a function of the observed weather patterns and combine information from different sources to capture the nuances of each individual situation.

A

The metrics over lead times for various approaches

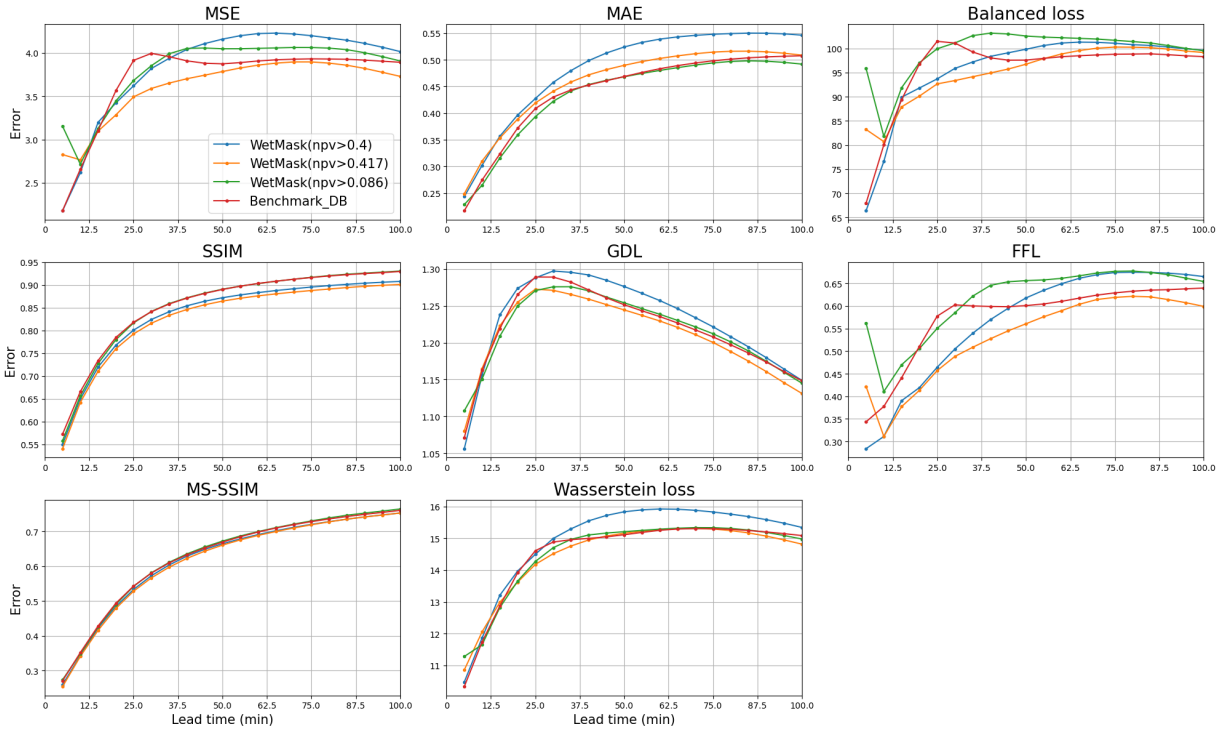

Figure A.1: The different metrics over the lead times with wet masks

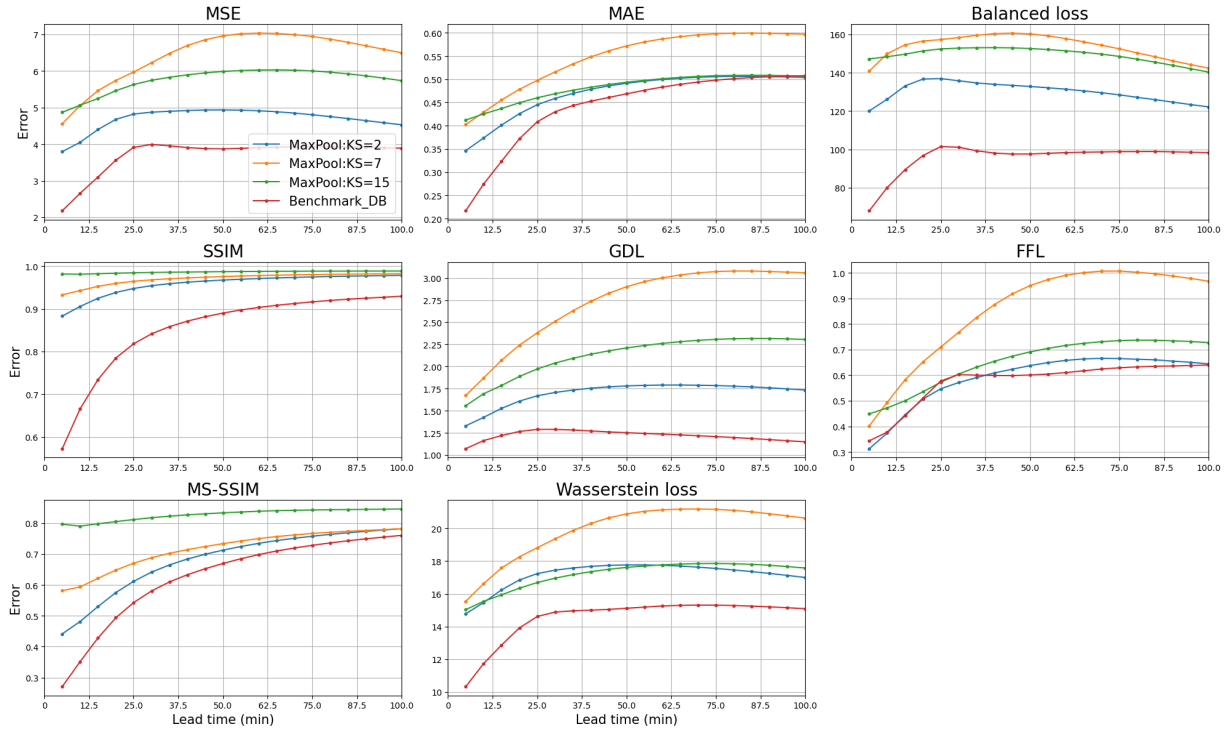

Figure A.2: The different metrics over the lead times when using max pooling operation in loss function

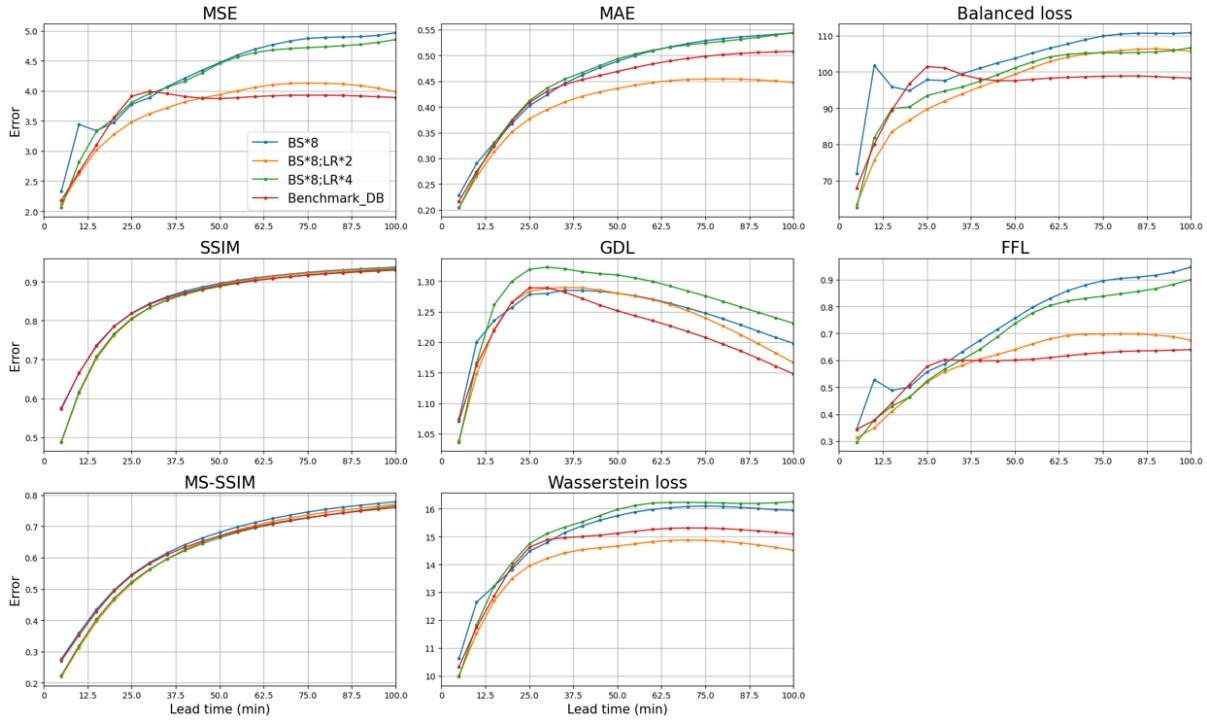

Figure A.3: The various metrics across different lead times in experiments conducted with larger batch sizes and learning rates

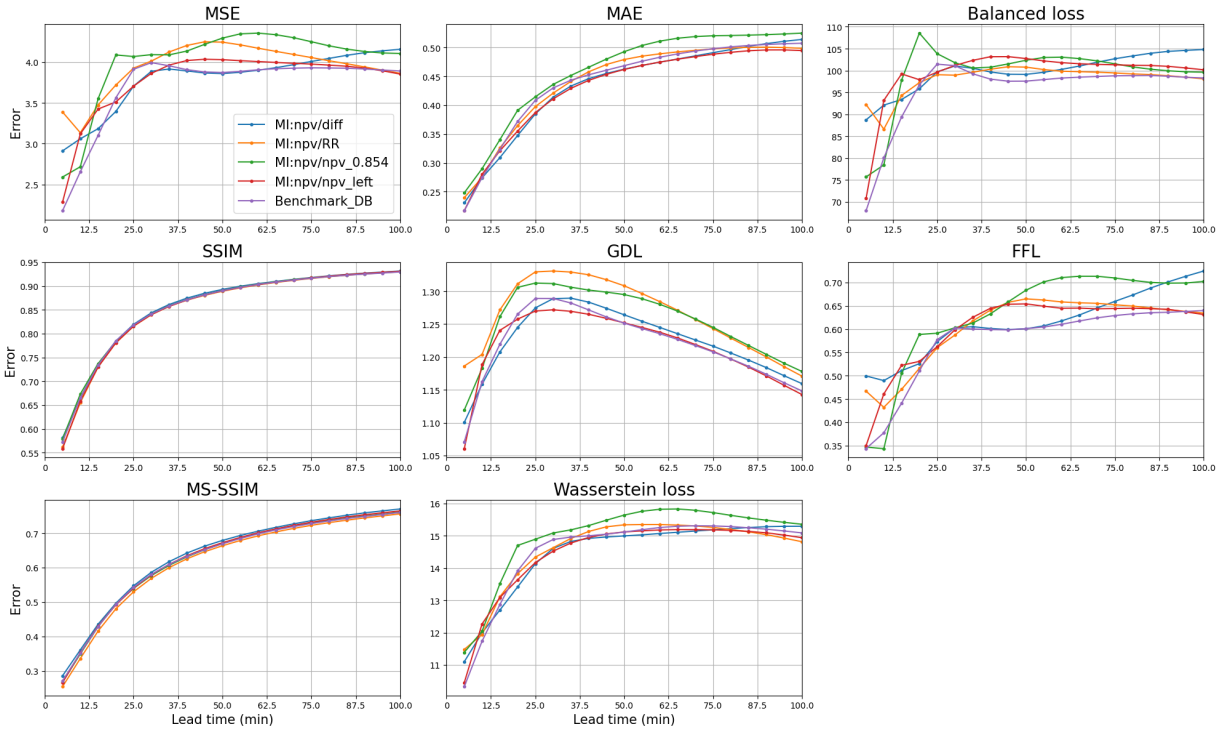

Figure A.4: The different metrics over the lead times, with additional model inputs

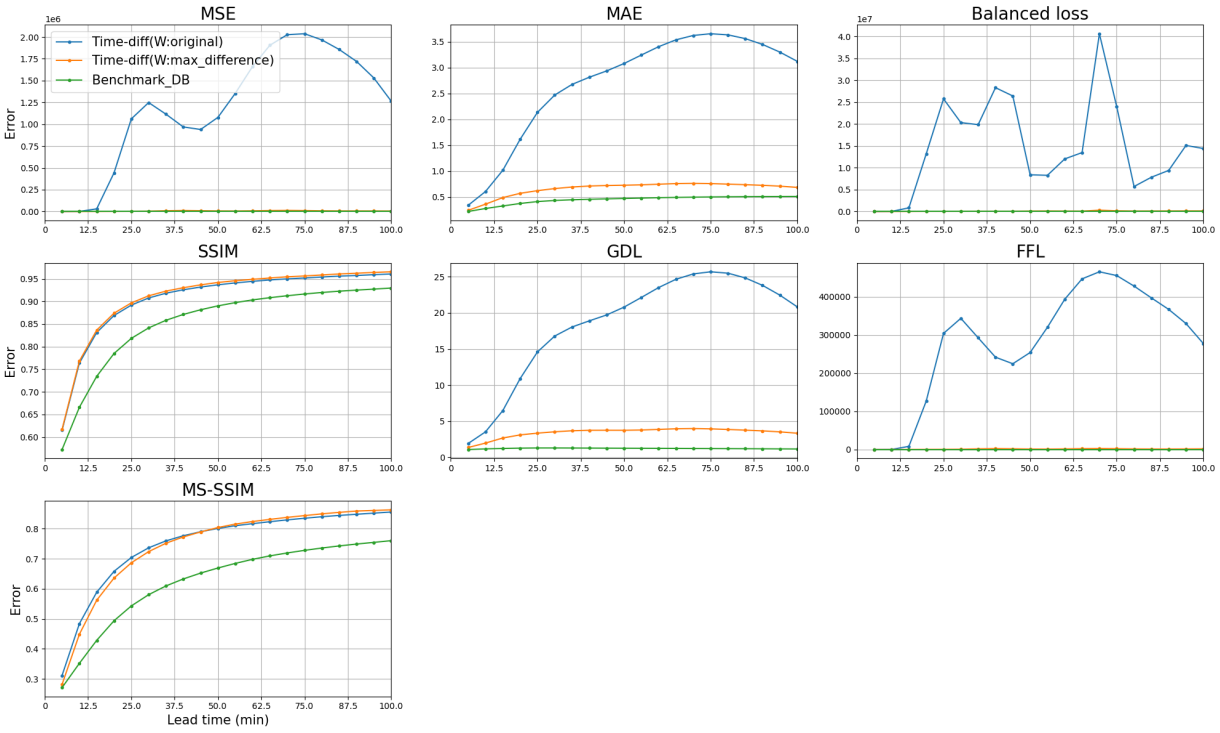

Figure A.5: The different metrics over the lead times, in the experiments with time-differenced data

B

Single model input: Modified radar images

In contrast to Section 3.4.2 in which we integrated additional datasets with radar images in NPV, these modified datasets were also evaluated as standalone inputs to replace NPV images. These supplementary datasets align with those depicted in Figure 3.8. As previously mentioned, these datasets, transformed from radar images in NPV, offer enhanced spatial and temporal information within rain fields. Gradient images depict variations and contrasts between adjacent pixels, while time-differenced images reveal trends about rain growth and decay at the pixel level. It's important to note that these datasets do not directly represent precipitation intensity. The resulting predictions are displayed in Figure B.1.

Predictions using gradient and time-differenced datasets appear slightly less sharp than those using NPV and RR images, which directly represent rain distribution. As the lead time increases, all predictions consistently exhibit a gradual increase in blurriness. It's worth noting that even without the input of precipitation intensity, models can effectively learn about actual rainfall from a single discrete dataset and generate accurate predictions. This type of experiment encourages further exploration with different datasets.

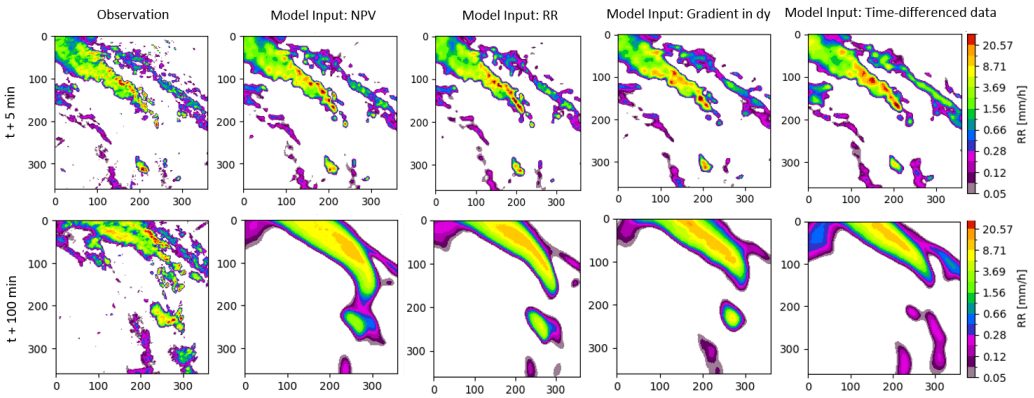

Figure B.1: Observation and predictions trained by different datasets which are transformed from radar images in NPV.

C

Two-stage loss optimization

In the initial phase, we conducted experiments involving two different loss functions at different training stages. When the Benchmark used a balanced loss, the loss value began to stabilize after 30,000 iterations, indicating that the model had reached the minimum, and further training was unlikely to significantly enhance its performance. Consequently, in this kind of experiment, another loss function, different from the initial one, was implemented in the subsequent stage. By sequentially employing these two different loss functions, we assumed the model inherits the optimizer state from the previous stage and refines its performance from other perspectives.

Table C.1 presents the changes in the experimental setup. In Experiment 1, during the first 40,000 iterations, the model was trained with the balanced loss. In the subsequent iterations, the target loss was switched to GDL, aiming to sharpen the predictions with more gradients. In Experiment 2, during the initial 40,000 iterations, the model's evaluation was based on the average performance of 20 outputs. In the following iterations, as the predictions in longer lead times were less sharp than those in the initial lead times, we directed the model's focus towards improving the predictive performance of the last 10 outputs.

Table C.1: Stage-base training: dividing the training process into two stages, each with its own specific loss function.

Iteration	Experiment 1	Experiment 2
0 - 40k	Balanced Loss	All 20 output frames
40k - 85k	GDL	Last 10 output frames

By looking at the loss curve and visual predictions (Figure C.1), we can observe a distinct shift in the loss occurring at the 40,000-iteration mark. In Experiment 1, GDL value continued to decrease after it was set as the target loss, eventually converging to around 0.45. However, this transition corresponded to a noticeable decline in visual predictions across all lead times, exacerbating the fading of rain. The visual results obtained with GDL in the second stage resembled the predictions reported in Dekker (2022), characterized by an overall underestimation of radar images.

In Experiment 2, after the initial 40,000 iterations, the loss was calculated with the last 10 outputs, causing an increase in loss value (Figure C.2). This shift was attributed to the fact that the model's performance in longer lead times was inferior to that in the earlier lead times. Visual predictions indicated a trade-off between the performance of the first half of the outputs and the latter half. When the focus of the loss shifted to the latter half of outputs, the predictions for the initial lead time gradually deteriorated, but there was no expected improvement in the longer lead times.

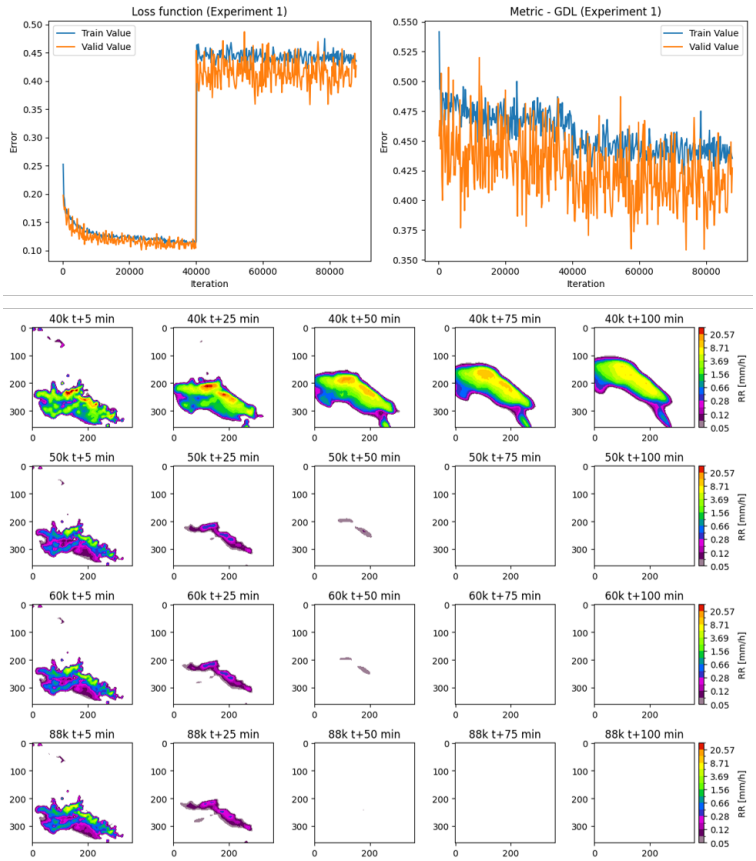

Figure C.1: Top: The track of the training process with different metrics. Bottom: Predictions after 40k iterations, where the loss function transitioned from the balanced loss to GDL at the 40k iteration.

In experiment 1, balanced loss emphasizes the intensity accuracy at the pixel level, while GDL penalizes the gradient difference between pixels. Such different principles of loss functions can be the reason for prediction degradation. The concept behind progressive training is one loss function focuses on capturing fundamental features of the data while the other further refine specific aspects. It is recommended that there should be a logical connection between two loss functions. In addition, when using two loss functions, you may need to adjust hyperparameters such as the learning rate for each loss function separately. The learning rate wasn't adjusted in our report, which could be a possible reason for the loss failing to decrease further in the second stage.

Both experiments were trained using pixel-wise loss functions. An intriguing idea is to incorporate a neighborhood loss function as the second loss during training. As mentioned in section 4.2.3, if the neighborhood loss function were the sole loss function, it might leave many pixels at zero, resulting in a checkerboard pattern. When used in the second stage, the neighborhood loss function is assumed to leave most pixels with their previous values while helping improve high-intensity predictions. However, time constraints prevented us from further exploring this idea.

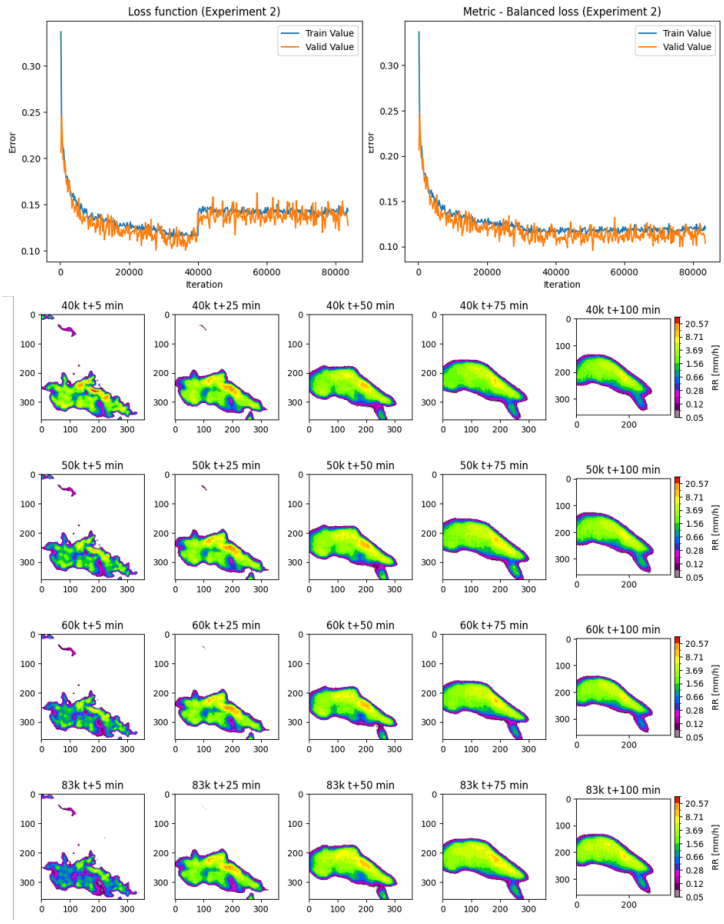

Figure C.2: Top: The track of the training process with different metrics. Bottom: Predictions after 40k iterations; before 40k iterations, the balanced loss averages 20 output frames, while in the second stage, it averages the latter 10 output frames.

D

The hidden circular pattern in low rain intensity

The colorbar in previous radar images only displayed pixel values greater than 0.417 NPV (equivalent to 0.05 mm/h). In this section, we aim to observe the pattern of smaller NPV values (0 – 0.417 NPV). In Figure D.1, the radar image discover distinct circular patterns when only depicting NPV values below 0.417. These circular patterns can not represent actual rain scenarios. Figure D.1 shows two circles, with one being more distinct and positioned outside the rainy area. Below it, there is another circle, but it partially overlaps with the main rainy region.

It is mentioned in Section 3.1 that KNMI operates two C-band Doppler weather radars in Herwijnen and Den Helder (Figure 3.1). These Doppler radars are likely responsible for the observed circular pattern, as their locations overlap each other. We have also observed this circular pattern in the forecast (Figure D.1). It's important to note that this circular pattern should not be predicted, as it results from the mutual interaction between the Doppler radars and raindrops.

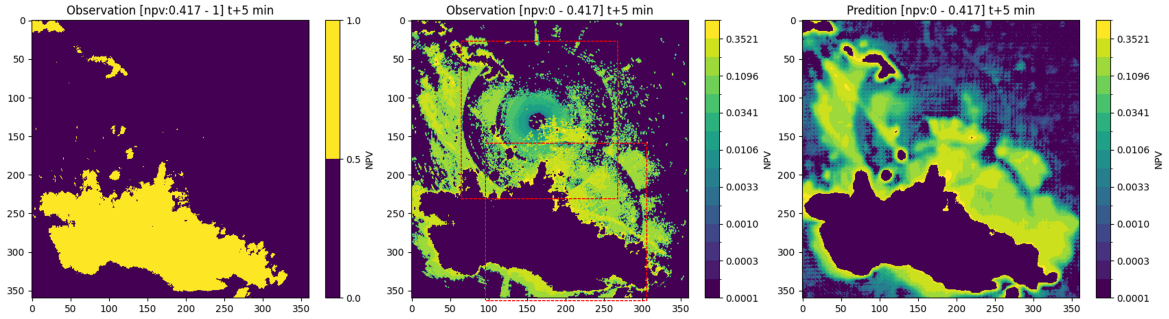

Figure D.1: Left: The binary image used to exhibit the rainy area, where NPV values between 0.417 and 1 are set to 1; Middle: Observation image filtering out NPV values greater than 0.417, with circular patterns marked with red squares; Right: Prediction image filtering out NPV values greater than 0.417. In our regular visual analysis, pixel values larger than 0.05 mm/h (0.417 NPV) will be displayed representing wet pixels.

In this thesis, our radar images typically focus on the rain field, displaying normalized pixel values greater than 0.417. However, the unprinted patterns suggest that the performance of the model may be influenced by Doppler weather radars. In most experiments, our models operate using NPV. As highlighted by Dekker (2022), training on NPV instead of RR places extra emphasis on low RR values due to their different dataset distributions. To prevent models from predicting these artifacts (circle patterns), it is suggested to either train models in the RR domain or perform data preprocessing in future work.

Bibliography

S. Agrawal, L. Barrington, C. Bromberg, J. Burge, C. Gazen, and J. Hickey. Machine learning for precipitation nowcasting from radar images, 2019.

G. Ayzel, T. Scheffer, and M. Heistermann. Rainnet v1.0: a convolutional neural network for radar-based precipitation nowcasting. *Geoscientific Model Development*, 13(6):2631–2644, 2020. doi: 10.5194/gmd-13-2631-2020. URL https://gmd.copernicus.org/articles/13/2631/2020/.

M. C. Bakkay, M. Serrurier, V. K. Burda, F. Dupuy, N. C. Cabrera-Gutierrez, M. Zamo, M.-A. Mader, O. Mestre, G. Oller, J.-C. Jouhaud, and L. Terray. Precipitaion nowcasting using deep neural network, 2022.

L. Balles, J. Romero, and P. Hennig. Coupling adaptive batch sizes with learning rates. 2017.

S. De, A. Yadav, D. Jacobs, and T. Goldstein. Big batch sgd: Automated inference using adaptive batch sizes, 2017.

D. Dekker. Perceptual losses in precipitation nowcasting. 2022.

F. Elsmann. Exploring the impact of echo top heights in generative models. 2023.

J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019.

E. Gilleland, D. Ahijevych, B. G. Brown, B. Casati, and E. E. Ebert. Intercomparison of spatial forecast verification methods. *Weather and Forecasting*, 2009.

S. Han, J. Pool, J. Tran, and W. J. Dally. Learning both weights and connections for efficient neural networks, 2015.

E. Hoffer, I. Hubara, and D. Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks, 2018.

I. Kandel and M. Castelli. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*, 2020.

C. Kaparakis and S. Mehrkanoon. Wf-unet: Weather data fusion using 3d-unet for precipitation nowcasting. *Procedia Computer Science*, 222:223–232, 2023. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2023.08.160. URL https://www.sciencedirect.com/science/article/pii/S1877050923009250. International Neural Network Society Workshop on Deep Learning Innovations and Applications (INNS DLIA 2023).

N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. 2017.

J. Ko, K. Lee, H. Hwang, S.-G. Oh, S.-W. Son, and K. Shin. Effective training strategies for deep-learning-based precipitation nowcasting and estimation. *Computers &amp Geosciences*, 161:105072, apr 2022. doi: 10.1016/j.cageo.2022.105072. URL https://doi.org/10.1016%2Fj.cageo.2022.105072.

R. Lagerquist and I. Ebert-Uphoff. Can we integrate spatial verification methods into neural-network loss functions for atmospheric science?, 2022.

H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets, 2017.

T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection, 2018.

D. Niu, H. Che, C. Shi, Z. Zang, H. Wang, X. Chen, and Q. Huang. A heterogeneous spatiotemporal attention fusion prediction network for precipitation nowcasting. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 16:8286–8296, 2023. doi: 10.1109/JSTARS.2023.3310361.

R. Prudden, S. Adams, D. Kangin, N. Robinson, S. Ravuri, S. Mohamed, and A. Arribas. A review of radar-based nowcasting of precipitation and applicable machine learning techniques, 2020.

S. Ravuri, K. Lenc, M. Willson, D. Kangin, R. Lam, P. Mirowski, M. Fitzsimons, M. Athanassiadou, S. Kashem, S. Madge, R. Prudden, A. Mandhane, A. Clark, A. Brock, K. Simonyan, R. Hadsell, N. Robinson, E. Clancy, A. Arribas, and S. Mohamed. Skilful precipitation nowcasting using deep generative models of radar. *Nature*, 597(7878):672–677, sep 2021. doi: 10.1038/s41586-021-03854-z. URL https://doi.org/10.1038%2Fs41586-021-03854-z.

G. S. B. Reichstein, M.and Camps-Valls. Deep learning and process understanding for data-driven earth system science. *Nature*, 2019.

C. J. Shallue, J. Lee, J. Antognini, J. Sohl-Dickstein, R. Frostig, and G. E. Dahl. Measuring the effects of data parallelism on neural network training. 2019.

X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. WOO. Convolutional lstm network: A machine learning approach for precipitation nowcasting. 28, 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/07563a3fe3bbe7e3ba84431ad9d055af-Paper.pdf.

X. Shi, Z. Gao, L. Lausen, H. Wang, D.-Y. Yeung, W. kin Wong, and W. chun Woo. Deep learning for precipitation nowcasting: A benchmark and a new model. 2017.

S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le. Don't decay the learning rate, increase the batch size, 2018.

Q. Tran and S. Song. Computer vision in precipitation nowcasting:applying image quality assessment metrics for training deep neural networks. *atmosphere*, 2019.

E. van der Kooij. Nowcasting heavy precipitation in the netherlands: a deep learning approach. 2021.

Y. Yang and S. Mehrkanoon. Aa-transunet: Attention augmented transunet for nowcasting tasks, 2022.

X. You, Z. Liang, Y. Wang, and H. Zhang. A study on loss function against data imbalance in deep learning correction of precipitation forecasts. *Atmospheric Research*, 281:106500, 2023. ISSN 0169-8095. doi: https://doi.org/10.1016/j.atmosres.2022.106500. URL https://www.sciencedirect.com/science/article/pii/S0169809522004860.

G. Zhang, L. Li, Z. Nado, J. Martens, S. Sachdeva, G. E. Dahl, C. J. Shallue, and R. Grosse. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model, 2019.