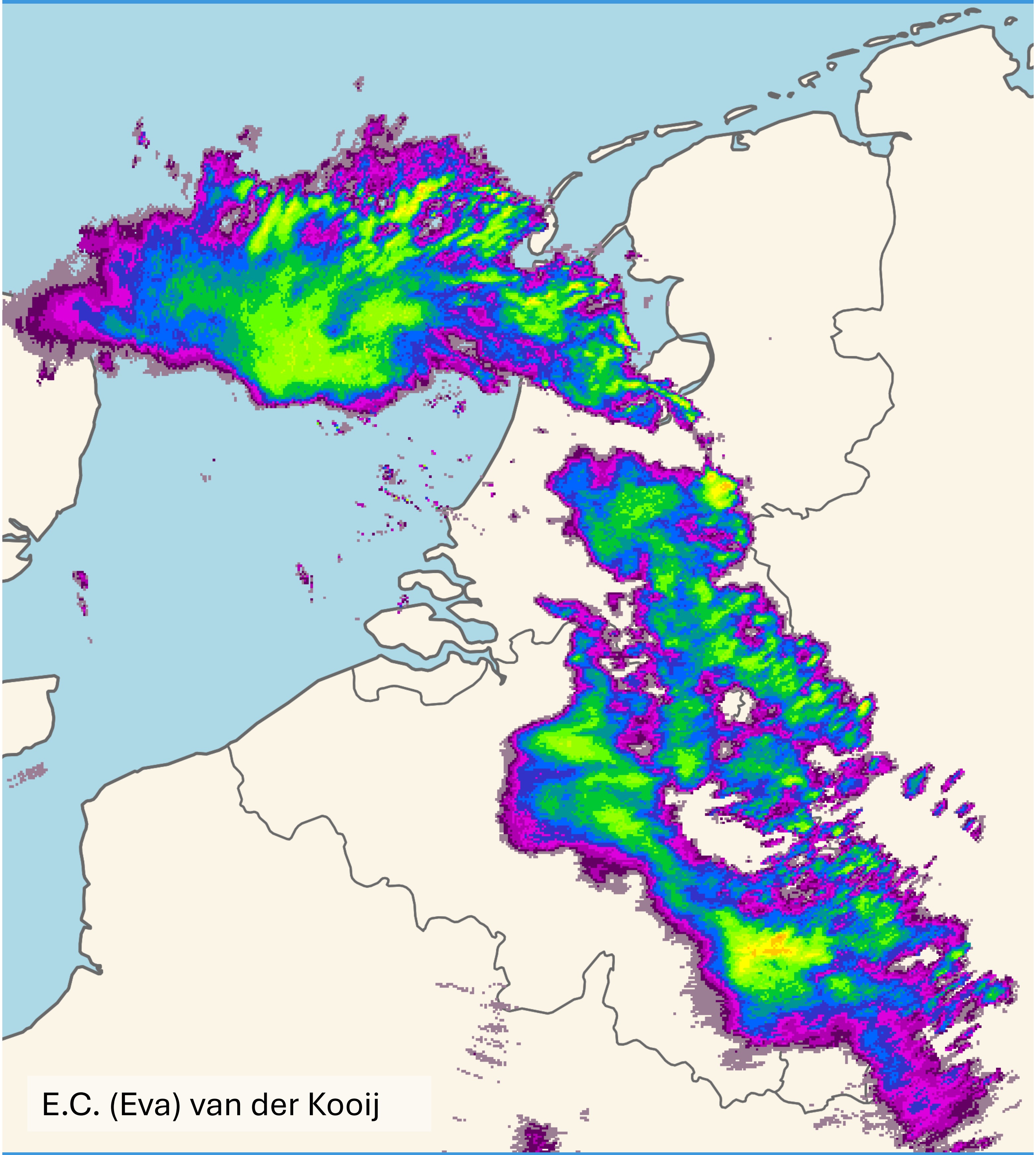
Nowcasting heavy precipitation in the Netherlands: a deep learning approach



E.C. (Eva) van der Kooij

Nowcasting Heavy Precipitation in the Netherlands: a Deep Learning Approach

by

E.C. (Eva) van der Kooij

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday July 6, 2021 at 12:00 AM.

Student number:	4306694	
Project duration:	September 1, 2020 – July 6, 2021	
Thesis committee:	Dr. M.A. (Marc) Schleiss,	TU Delft, dept. Geoscience & Remote Sensing (CiTG), Chair
	Dr. F. (Francesco) Fioranelli,	TU Delft, dept. Microelectronics (EEMCS)
	Dr. R. (Riccardo) Taormina,	TU Delft, dept. Watermanagement (CiTG)
	Ir. D. (Dorien) Lugt,	HKV Lijn in Water, Company supervisor
	Dr. M. (Mattijn) van Hoek,	HKV Lijn in Water

Preface

In this report I present the results of my master thesis research, which I conducted between September 2020 and July 2021. This research was executed at HKV Lijn in Water as a graduate intern, with help from the Royal Dutch Meteorological Instute (KNMI). This project finalizes my master of Environmental Engineering and at the same time marks the end of my time as a student at TU Delft. This has been a period in which I have learned a lot, both academically and personally, and I would like to express my gratitude to everyone who has been a part of this.

I want to thank Dorien Lugt and Mattijn van Hoek, my supervisors from HKV, for the support they have given me throughout this year. Thanks for giving me some guidance into the world of nowcasting and machine learning and for providing me with the right tools to learn the skills I needed to conduct this research. Our weekly meetings were very helpful but also very 'gezellig' and a good substitute for what coffee breaks could have been if we had been at the office together more often. I would also like to thank my TU Delft supervisors Marc Schleiss, Riccardo Taormina and Francesco Fioranelli. To Marc, the chair of my committee, thanks for the truly extensive amount of support. Our meetings always provided me guidance and left me energized and eager to keep on working. To Riccardo, thanks for the support with respect to the machine learning part of this research. When I started this project this was something I had no experience with, and our meetings have helped me to be able to dive deep into this subject. To Francesco, thank you for the extensive feedback on draft reports and presentations and for always sending me notes you made during meetings with the committee. To Aart Overeem and Hidde Leijnse, from KNMI, for helping me get familiar with the radar data and providing outside perspective.

I would also like to express my gratitude to my family and friends for the mental support during this period. Writing an entire thesis from my bedroom was not how I had envisioned this period when I started my masters and it was definitely challenging at times. To Stijn and Roos, my roommates, thanks a lot for all the oat milk cappuccinos, lunchtime dance parties and necessary complaining sessions. To my family, for always encouraging me. To Jolijne and Fréderike, my closest friends, for always picking up the phone, going on walks with me and getting me out of my research bubble.

E.C. (Eva) van der Kooij
Rotterdam, June 2021

Abstract

Accurate short-term forecasts, also known as nowcasts, of heavy precipitation are desirable for creating early warning systems for extreme weather and its consequences, e.g. urban flooding. In this research, we explore the use of machine learning for short-term prediction of heavy summer rainfall showers in the Netherlands.

We explore the use of a recurrent, convolutional neural network (TrajGRU, Shi et al., 2017) with lead times of up to 100 minutes. We aim to train the model to predict the formation and dissipation of dynamic, heavy, localized rain events, a task for which traditional Lagrangian extrapolation based nowcasting methods still come up short. The network is trained, validated and tested on a 13-year archive of radar images with 5-min temporal and 1-km spatial resolution from the precipitation radars of the Royal Netherlands Meteorological Institute (KNMI).

We report on different ways to optimize predictive performance for heavy precipitation events through two experiments. In the first experiment different training dataset compositions are explored. The large dataset available provides many possible configurations for training. To focus on heavy rainfall intensities, we use different subsets of the radar dataset through using different conditions for event selection and varying the ratio of light and heavy precipitation events present in the training data set. In the second experiment we change the loss function used to train the model.

To assess the performance of the model, we compare our method to a current state-of-the-art deterministic Lagrangian extrapolation-based nowcasting system from the pySTEPS library, S-PROG (Seed et al, 2003). The results of the experiments are used to discuss the pros and cons of machine-learning based methods for precipitation nowcasting and possible ways to further increase performance.

Model behaviour was found to be significantly influenced by the formulation of the loss function. It was concluded that there is always a trade-off between performance at low rainfall intensities and performance at high rainfall intensities: (1) If a model makes smaller errors at low rainfall intensities this results in a low total error, but also in the failure to detect high rainfall intensities. (2) If model performance is improved at detecting high rainfall intensities, this results in a decreased performance at low rainfall rates and increases the total error.

Acronyms

ANN Artificial Neural Network.

CNN Convolutional Neural Network.

CSI Critical Succes Index.

DL Deep Learning.

FAR False Alarm Ratio.

FSS Fractions Skill Score.

GRU Gated Recurrent Unit.

LR Learning Rate.

LSTM Long Short-Term Memory.

MAE Mean Absolute Error.

MSE Mean Square Error.

NWP Numerical Weather Prediction.

OF Optical Flow.

POD Probability Of Detection.

QPE Quantitative Precipitation Estimation.

QPF Quantitative Precipitation Forecast.

RMSE Root Mean Square Error.

RNN Recurrent Neural Network.

Glossary

lead time Forecast horizon; the length of time into the future for which predictions are prepared.

optical flow The pattern of apparent motion of objects in consecutive images..

reflectivity Expression for the amount of backscattered radiation measured by a radar.

List of Figures

List of Tables

Contents

1 Introduction

Nowcasting is defined as short term (0-6 hours) quantitative precipitation forecasting. Accurate nowcasts are desirable for creating reliable early warning systems for extreme weather and its consequences, e.g. urban flooding. With the projected increase in frequency and intensity of precipitation extremes (KNMI, 2014; Lenderink et al., 2011) this becomes even more important.

Quantitative precipitation forecasts (QPFs) are traditionally generated in one of two ways: with numerical weather prediction (NWP) models or with statistical extrapolation of radar observations. NWP models simulate the physical state of the atmosphere by numerically solving a set of partial differential equations that govern the physical state of the atmosphere and by parameterizing small scale (subgrid) processes, bounded by a set of initial conditions and boundary conditions. This can produce reliable forecasts but it is a computationally expensive process and it generates forecasts at coarse temporal and spatial scales and with low update rates due to long spin-up times (the time it takes for the model to reach physical equilibrium). Lots of improvements have been made in this field with for example data assimilation for lower calculation times and increased accuracy, and higher spatial and temporal resolution (Sun et al., 2014). However, NWP models are still outperformed by extrapolation-based methods in 0-2 h forecasting range (Simonin et al., 2017; Sun et al., 2014). This is the reason that for nowcasting in this time range, radar extrapolation-based models are used.

Extrapolation-based methods predict the evolution of the rainfall process based on the last radar observations. These models are based on the fundamental assumption that both the rainfall intensity and the motion fields are stationary: rain cells are assumed to only evolve slowly over time and move along a principal direction of motion. This concept is commonly referred to as ”Lagrangian persistence” (Germann & Zawadzki, 2002). This type of forecast consists of two steps. (1) Tracking: Determine the motion field based on the most recent observations with for example an optical flow (OF) algorithm. (2) Extrapolation: Application of an advection scheme to extrapolate the last observation along the direction of the flow. In Figure 1.1 this process is illustrated.

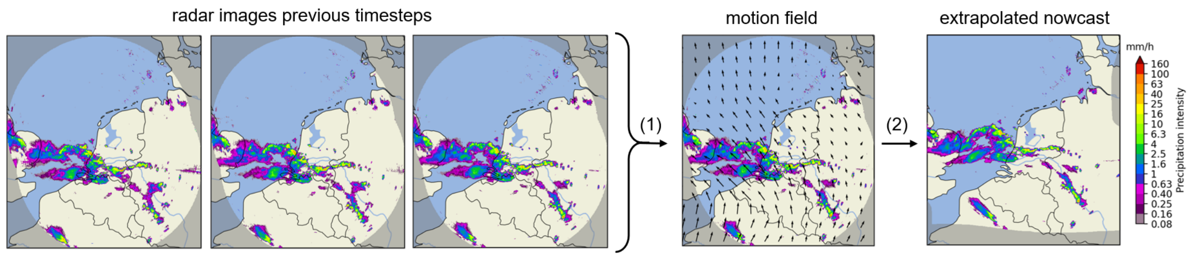


Figure 1.1: An example of an extrapolation-based nowcast at t_0 = 2020-08-16 17:00. In step (1) the motion field is determined from previous radar observations, in step (2) the last observation is extrapolated with an advection scheme.

These nowcasts can also be made in a probabilistic sense, by adding perturbations in the forecasting process, generating an ensemble of nowcasts. Extrapolation-based models, with varying methods for tracking and extrapolation, are employed in many operational nowcasting systems (Reyniers, 2008; Woo & Wong, 2017). Many studies have been done on the verification of these models in various regions of the world (e.g. Foresti et al., 2016; Woo & Wong, 2017). In the Netherlands, Imhoff et al. (2020) performed an analysis of model performance of several different extrapolation-based models. Skillful lead times (the time span a forecast is deemed useful) between 25 minutes, up to almost 2 hours were found and model performance was found to be worst for summer precipitation events.

Uncertainty in extrapolation-based nowcasts can originate from three sources (adapted from Germann et al. (2006) and Foresti et al. (2019)): (1) Uncertainty in the radar derived quantitative precipitation estimation (QPE, refers to the rainfall field), (2) The assumption of Lagrangian persistence does not hold because either the rainfall field is not stationary (e.g. due to dissipation, intensification or initiation of rainfall), or the motion field is not stationary; (3) Model errors in the implementation of Lagrangian persistence (e.g. inaccuracies in motion field retrieval or extrapolation scheme application). Improvement in these types of nowcasts can be made by reducing these uncertainties. For example, by blending with NWP (Chung & Yao, 2020; Nerini et al., 2019; Sun et al., 2014) , data assimilation, e.g. using observed or model derived wind variables (Bechini & Chandrasekar, 2017; Winterrath & Rosenow, 2007) and by trying to model growth and decay processes in the rainfall field (Foresti et al., 2019).

In recent years, several studies have also explored the potential of deep learning for nowcasting. With deep learning, no assumptions have to be made and no explicit rules are programmed. A model architecture is provided and the model parameters are determined by the deep learning process. So in contrary to extrapolation-based models, the underlying processes don't need to be understood and described. You let the deep learning model search for the relationship between input and output based on historical data. By using deep learning for nowcasting, the hope is to overcome the assumptions that are made in extrapolation-based models, like Lagrangian persistence, and hope that the model can capture more complex processes, like intensification and dissipation of the rainfall field, as well.

One of the first successful implementations of deep learning for rainfall nowcasting can be attributed to Shi et al. from the Hong Kong Meteorological Observatory with his introduction of ConvLSTM (2015) and in a follow up study with TrajGRU (2017), two convolutional, recurrent neural networks. Both outperformed extrapolation based model ROVER (Real-time Optical flow by Variational methods for Echoes of Radar), the currently operational nowcasting system employed in the Hong Kong area. Since then, many more studies have been published on the topic of deep learning for precipitation nowcasting. Notable studies included research into fully convolutional networks (Ayzel et al., 2020) , different type memory blocks (Cao et al., 2019; Wang et al., 2019), the use of attention mechanisms (Yan et al., 2020), model stacking (Franch et al., 2020), multi-channel radar input (Jing et al., 2019; Tran & Song, 2019), adding real-time meteorological re-analysis data (Song et al., 2019; W. Zhang et al., 2019), the use of vanilla recurrent neural networks (RNNs) to reduce the number of parameters (Singh et al., 2017b) and adversarial training (Jing et al., 2019; Singh et al., 2017a; Tian et al., 2019). Although many successes have been achieved in terms of low forecast errors, some recurring weaknesses have been noted. Deep learning models tend to produce excessively smoothed rainfall fields and consistently underestimate high rainfall values (e.g. Ayzel et al., 2020; Franch et al., 2020; Marrocu & Massidda, 2020).

1.1. Research objective

As mentioned, summer precipitation events were found to have worse extrapolation-based model performance compared to events in other seasons (Imhoff et al., 2020). Summer precipitation is often caused by convection and is of a more dynamic nature, and the assumption of Lagrangian persistence applies to a lesser degree. These type of events are however the heaviest and prone to cause problems, like flooding. It is thus important to improve nowcasts for this type of event.

This research is an exploration how deep learning can be used to predict heavy summer precipitation in the Netherlands with lead times up to 100 minutes. The goal of this research is not to produce the best possible model, but to investigate how our choices in model training setup influence the model's ability to accurately predict heavy summer rainfall events specifically. To that end, different strategies for training a deep learning model based on TrajGRU (Shi et al., 2017) were explored. Model performance was compared to an extrapolation-based benchmark, S-PROG (Spectral-PROGnosis) as introduced by Seed et al. (2003). A 13-year long archive of radar reflectivity images from the Royal Netherlands Meteorological Institute (KNMI) was available for training, validation and testing. A method to extract (heavy) precipitation events from this archive is presented. The nowcasts were verified using several metrics. Both average model performance and event-by-event based model performance are analysed.

1.2. Research questions

The research questions of this master thesis project are as follows:

1. How can a deep, recurrent, convolutional neural network, be trained to nowcast heavy rainfall events in the Netherlands, using a large radar reflectivity composites archive?
2. How does this model compare to S-PROG, a state-of-the-art extrapolation based nowcasting system?

To answer research question one, different training set-ups for the deep learning model are employed, with the aim to emphasize heavy precipitation. Two experiments were conducted: (1) comparison of different training datasets: dataset size and ratio of heavy precipitation events are varied, and (2) comparison of different loss functions. To answer research question two, the deep learning models that were created, are compared to the extrapolation-based benchmark S-PROG.

In Chapter 2, background information is provided on the radar reflectivity dataset and the models that are used in this research. In Chapter 3, the methodology is explained. In Chapter 4, the results are presented and interpreted. Results are discussed and compared to related work in Chapter 5, and conclusions and recommendations are given in Chapter 6.

2 Background

In this chapter, background information is provided on extreme precipitation in the Netherlands. The radar extrapolation-based model S-PROG and deep learning model TrajGRU are introduced, including some deep learning principles that are needed to understand the workings of TrajGRU.

2.1. Extreme precipitation in the Netherlands

In the Netherlands, extreme precipitation can be caused by two mechanisms: frontal precipitation and convective precipitation. The mechanisms are illustrated in Figure 2.1. Frontal precipitation occurs when two air masses of different temperatures meet. The cold, dense air will cause the warm, less dense air to rise over the cold air, which causes the air to cool and decrease its ability to hold moisture. This causes the moisture in the air to condensate and form clouds, which leads to precipitation. When warm air moves towards a cold air mass (a warm front), this leads to longer periods of low intensity rainfall. When cold air moves towards a warm air mass (a cold front), this leads to shorter periods of higher intensity rainfall.

Convective precipitation is the result of convection: the upwards motion of air due to warming of the Earth's surface by the sun. When the air near the surface warms, it becomes less dense and starts to rise. This will cause the air to cool and decrease its ability to hold moisture, causing condensation. Convective rain showers are typically of high intensity and short duration.

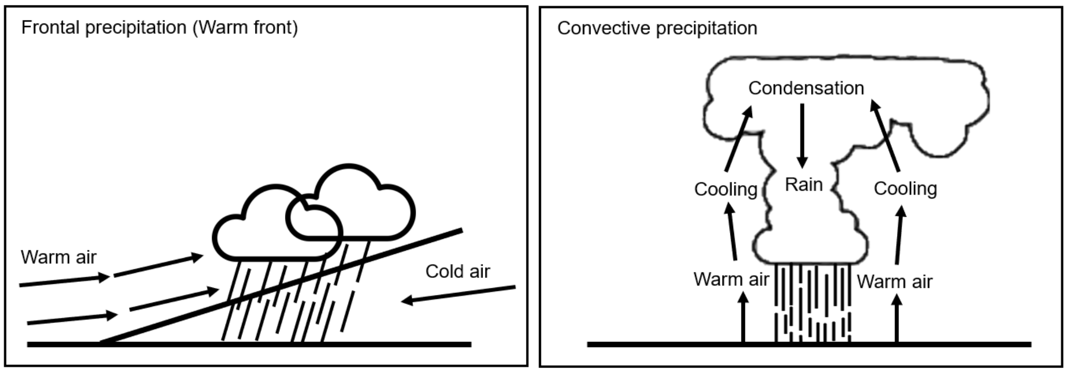


Figure 2.1: Diagrams illustrating the mechanisms of frontal precipitation formation and convective precipitation formation.

Frontal precipitation is responsible for precipitation extremes in winter. It is easier to predict because of longer lifespans and less vertical development. Convective precipitation is responsible for most precipitation extremes in summer. It is much harder to predict because it's a smaller scale process, both spatial and temporal, and vertical development plays a large role.

Increased frequency and intensity of precipitation extremes have been observed over the past century (Buishand et al., 2013) and this trend has been projected to continue. Because of increased temperature due to climate change, the atmosphere can hold more moisture, resulting in precipitation extremes occurring more often and with higher rainfall rates (KNMI, 2014; Lenderink et al., 2011).

2.2. KNMI radar reflectivity dataset

Radar is an acronym for RAdio Detection And Ranging. A radar emits pulses of electromagnetic radiation with a wavelength between 1-10 cm and measures the backscattered radiation at the emitted wavelength. In meteorology, radars are used for the observation of precipitation. Rainfall rates can be derived from the amount of backscattered radiation, expressed as radar reflectivity z. Since z spans over a large range, reflectivity is expressed on a logarithmic scale in unit dBZ. The relationship between reflectivity and rainfall rate used in this research is one defined by Marshall et al. (1955), see (2.1), with reflectivity Z in dBZ and rainfall rate R in mm/h.

$$Z = 200R^{1.6} \tag{2.1}$$

Although the quality of the radar reflectivity measurements and the conversion to rainfall rates are important factors in making accurate nowcasts, this is outside the scope of this research. For further information on radar meteorology, see Rauber and Nesbitt (2018).

The data used in this research are radar reflectivity composites produced by the Royal Netherlands Meteorological Institute (KNMI). KNMI operates two C-band Doppler weather radars, previously in De Bilt and Den Helder, and in 2016/2017 replaced by new radars in Herwijnen and Den Helder. See figure 2.2 for the locations and ranges of the radars. The radars operate at a 5.3 cm wavelength. Multiple products derived from the radars measurements are available, among which quantitative precipitation estimations (QPE) that are validated with ground observations. For the purpose of this research it is important that data is available in near real-time, so the low level processed, unvalidated radar reflectivity composites at 1500 m height were used.

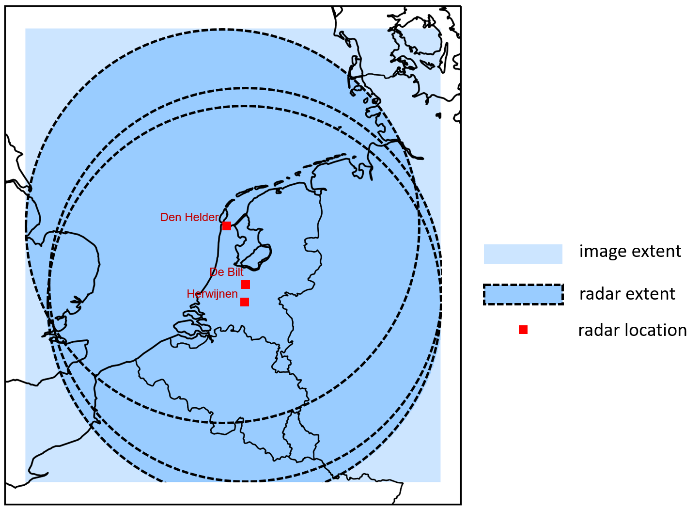


Figure 2.2: Weather radar network of the Royal Netherlands Meteorological institute, old locations (Den Helder and De Bilt) and new locations (Den Helder and Herwijnen).

Radar reflectivities are measured by both radars in scans with elevation angles of 0.3, 1.1, 2.0, and 3.0 degrees. From the volumetric scans performed by the radars, horizontal cross sections of reflectivity at 1,500 m height are generated by linear interpolation. These images are then combined into one radar reflectivity composite by taking a weighted average of the radar reflectivities. The weight of a point is a function of the distance to the radar. The product has a 1 km x 1 km spatial resolution and a 5-minute temporal resolution. The data is provided in hdf5 files, which contain images (700x765 pixels 2D arrays) of the 8-bit (range 0-255) radar reflectivity values in 0.5 dBZ intervals. Where no reflectivity measurements are available, outside the range of the radars or when one of the radars wasn't working, a no-data value of 255 is used. Transformation from 8-bit integer to reflectivity in dBZ is done with the following formula: Z = (pixel value * 0.5) - 32.0.

The described product has been archived since 2008, so over 13 years of data was available for this research. The radar reflectivity composites can be retrieved through the KNMI data platform via an API. The product name of the archive is `radar_tar_refl_composites/versions/1.0`. The near real-time images are available after 3 to 4 minutes and have product name `radar_reflectivity_composites/versions/2.0`.

2.3. Benchmark: S-PROG

In the Netherlands, KNMI has recently employed STEPS (Bowler et al., 2006) as the new operational nowcasting system. This is a probabilistic, radar extrapolation-based nowcasting method. This research focuses on deterministic nowcasts, so the deterministic extrapolation-based model S-PROG, on which STEPS was based, was chosen as the benchmark. S-PROG is short for Spectral PROGnosis and was introduced by Seed (2003). In this research the model was implemented using pySTEPS (Pulkkinen et al., 2019), an open-source Python framework for nowcasting systems.

S-PROG is an optical-flow advection based nowcasting system that also takes the temporal evolution of the rainfall field into account. The estimation of the evolution of the rainfall field is based on the property that every rainfall field contains features at different spatial scales, which develop at different temporal scales. Smaller features evolve faster and bigger features evolve slower (Germann & Zawadzki, 2002). The precipitation field is disaggregated into j cascade levels, each cascade level representing a different spatial scale. For each cascade level, autoregressive parameters are determined in order to determine the temporal evolution at this scale. Each scale is extrapolated separately using the resulting autoregressive models. The final nowcast is then calculated by summing the advected Lagrangian forecasts at all cascade levels. In case of the pySTEPS implementation, the workflow is as follows:

1. Read radar composites. Values are transformed from dBZ to rainfall rate R in mm/h, and then log-transformed to dBR.
2. Determine of motion field with Lucas-Kanade method. Lucas-Kanade is an optical flow method that yields a dense motion field by computing the motion vector for every pixel (Lucas & Kanade, 1981). In Figure 2.3 an example of a motion field is presented.
3. Extrapolate the rainfall field using semi-Lagrangian advection.
4. Apply a spectral decomposition into j cascade levels with a fast Fourier transform (FFT) and a Gaussian band-pass filter. In Figure 2.4 an example of a decomposition into 7 cascade levels is presented.
5. Estimate the Lagrangian autocorrelation coefficients $\rho_{j,1}$ and $\rho_{j,2}$ for each cascade level.
6. Determine the AR(2) autoregressive model parameters $\phi_{j,1}$ and $\phi_{j,2}$ from estimated Lagrangian autocorrelation coefficients $\rho_{j,1}$ and $\rho_{j,2}$ using the Yule-Walker equations. The resulting autoregressive model is equation 2.2.

$$R_j(x,y,t) = \phi_{j,1}R_j(x,y,t-\Delta t) + \phi_{j,2}R_j(x,y,t-2\Delta t) \tag{2.2}$$

7. Calculate the forecast by summing the advected Lagrangian forecasts of each cascade level, thereby recomposing the rainfall field.

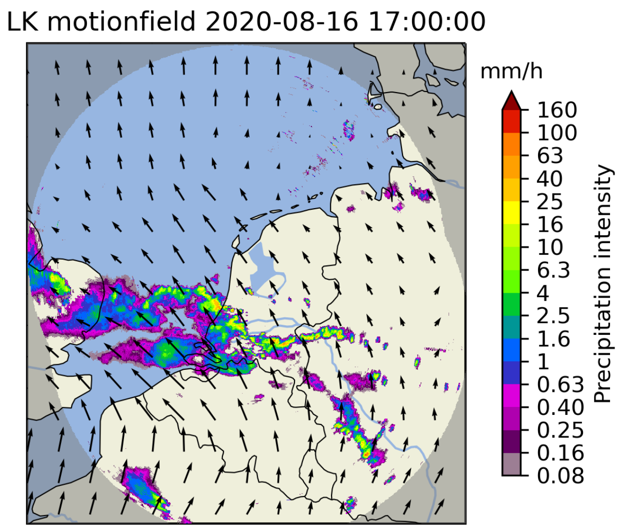


Figure 2.3: Motion field at t = 2020-08-16 17:00, derived with the Lucas Kanade method, from frames t-3, t-2, t-1 and t. The motion vector for every 50th pixel is shown.

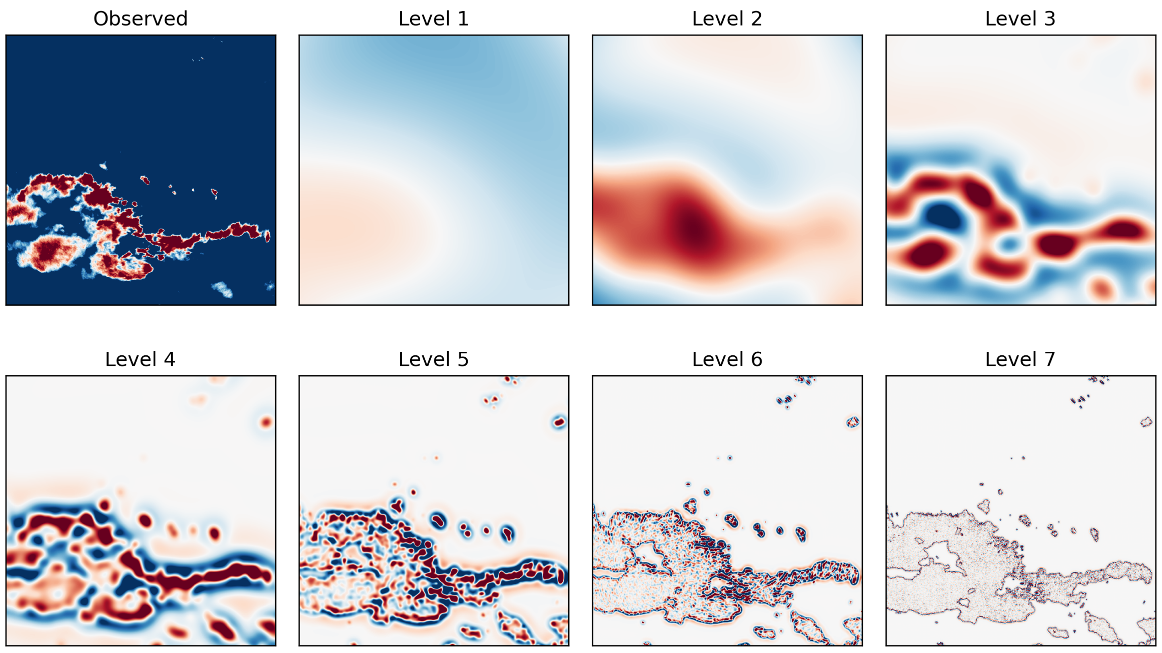


Figure 2.4: Precipitation field at t = 2020-08-16 17:00 and the resulting cascade levels from a decomposition into 7 levels with a fast Fourier transform using a Gaussion band-pass filter.

2.4. Deep learning: TrajGRU

Deep learning is the part of machine learning that concerns artificial neural networks with many layers and thus many model parameters. This makes these type of networks suited for modelling complex, highly non-linear processes. The model used in this research is a recurrent, convolutional neural network. In Appendix A, some of the fundamentals of these tools are explained. For more information on the fundamentals of deep learning, see A. Zhang et al. (2021).

One of the first successful implementations of deep learning for precipitation nowcasting was the Convolutional Long Short-Term Memory (ConvLSTM) as introduced by Shi et al. (2015), who approached precipitation nowcasting from a computer vision perspective and treated it as a video prediction problem. This resulted in a sequence-to-sequence (both input and output are spatiotemporal sequences) deep learning model using LSTMs (Long Short Term Memory) models for the temporal dependencies, but replacing the vector operations of a traditional LSTM with convolutional operations to capture spatial dependencies. In a follow-up study the Trajectory Gated Recurrent Unit (TrajGRU) was introduced (Shi et al., 2017), replacing the LSTM with a GRU, which is easier to train, and introducing a ”trajectory” component to learn location-variant structures.

A GRU is a mechanism in a recurrent neural network (RNN) that controls the information flow between inputs and outputs by applying gating mechanisms. The module contains a reset gate that controls what information from the previous inputs (hereafter referred to as the hidden state) should be kept and what should be forgotten, and an update gate which controls what information from the new input should be written to the new hidden state. This allows the model to have both a short term memory through the reset gate, and a long term memory through the update gate. For a more detailed explanation on GRUs see Appendix A.

Depending on the direction of flow, different locations in the new inputs are related to locations in the previous hidden state. Where the recurrent connections in ConvLSTM are fixed for every location, TrajGRU allows for dynamic recurrent connections. Specifically, the network employs a subnetwork γ in every GRU cell: a one-hidden-layer convolutional neural network that generates a motion field between the new input X_t and the previous hidden state H_{t-1}. A ’warp’ function then determines important neighbourhoods for each location at each time step, using these motion fields and the previous hidden state. These neighbourhoods are then used in the creation of the update gate Z_t, (Equation 2.4), the reset gate R_t (Equation 2.5) and the candidate hidden state $\tilde{H}_t$ (Equation 2.6), which control what information flows to the new hidden state. The activation function that is used in the calculation of the candidate hidden state is a Leaky ReLU with negative slope parameter 0.2. The new hidden state is calculated the same as in a regular GRU (Equation 2.7).

$$U_t, V_t = \gamma(X_t, H_{t-1}) \tag{2.3}$$

$$Z_t = \sigma(W_{xz} * X_t + \sum_{l=1}^{L} W_{hz}^l * \mathrm{warp}(H_{t-1}, U_{t,l}, V_{t,l})) \tag{2.4}$$

$$R_t = \sigma(W_{xr} * X_t + \sum_{l=1}^{L} W_{hr}^l * \mathrm{warp}(H_{t-1}, U_{t,l}, V_{t,l})) \tag{2.5}$$

$$\tilde{H}_t = f(W_{xh} * X_t + R_t \circ (\sum_{l=1}^{L} W_{hh}^l * \mathrm{warp}(H_{t-1}, U_{t,l}, V_{t,l}))) \tag{2.6}$$

$$H_t = (1 - Z_t) \circ \tilde{H}_t + Z_t \circ H_{t-1} \tag{2.7}$$

TrajGRU employs an encoder-forecaster structure. It downsamples three times in the encoder, and upsamples three times in the forecaster. The set-up used in this research takes 5 input images and yields 20 output images. Parameters like kernel size, stride, padding etc. are listed in Table 2.1 and visualized in Figure 2.6. A schematic overview of the complete model can be seen in Figure 2.5.

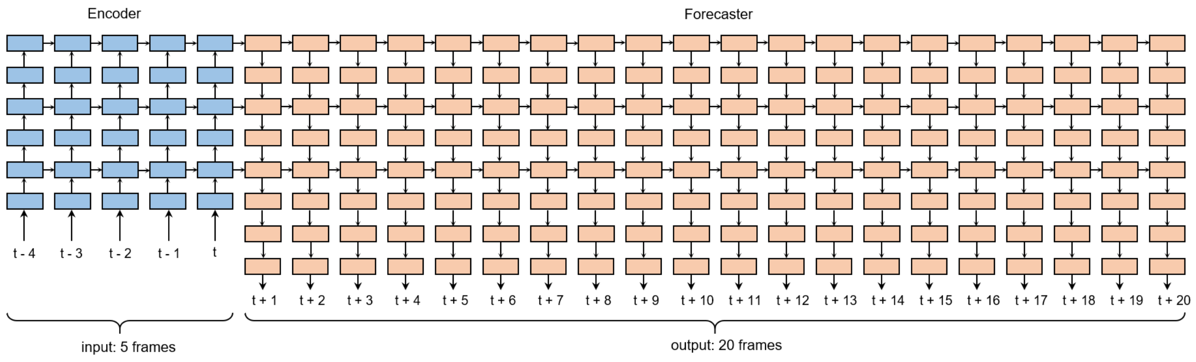


Figure 2.5: Model architecture of TrajGRU with 5 input frames and 20 output frames

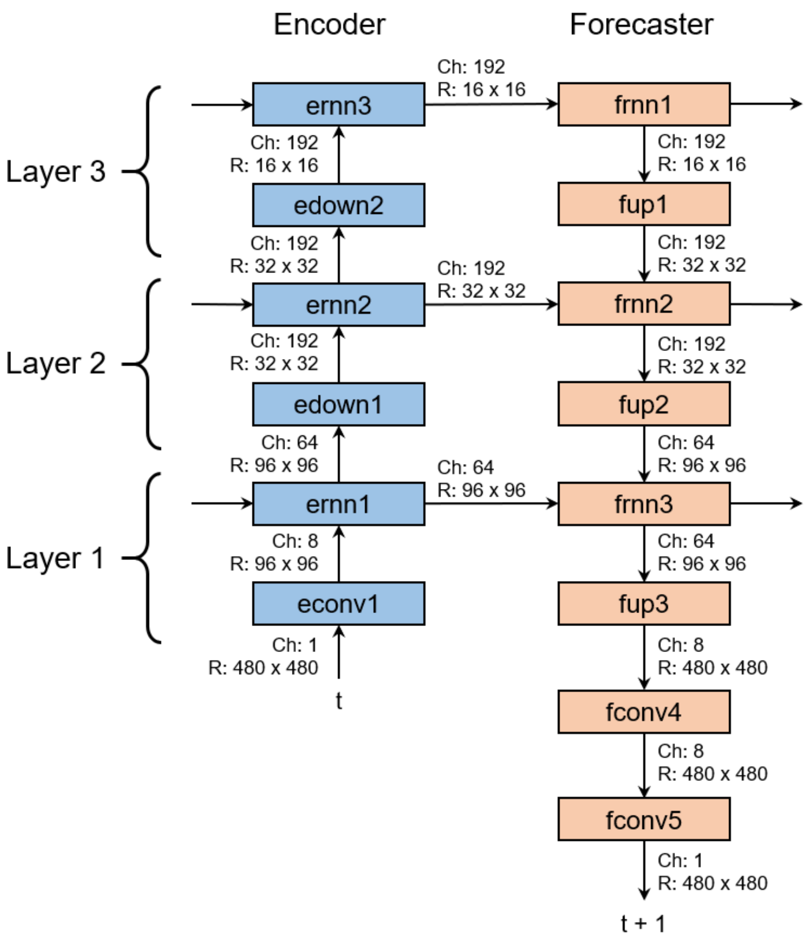


Figure 2.6: Close-up of TrajGRU model architecture

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	-

Table 2.1: Features of the TrajGRU architecture

3 Methodology

3.1. Process overview

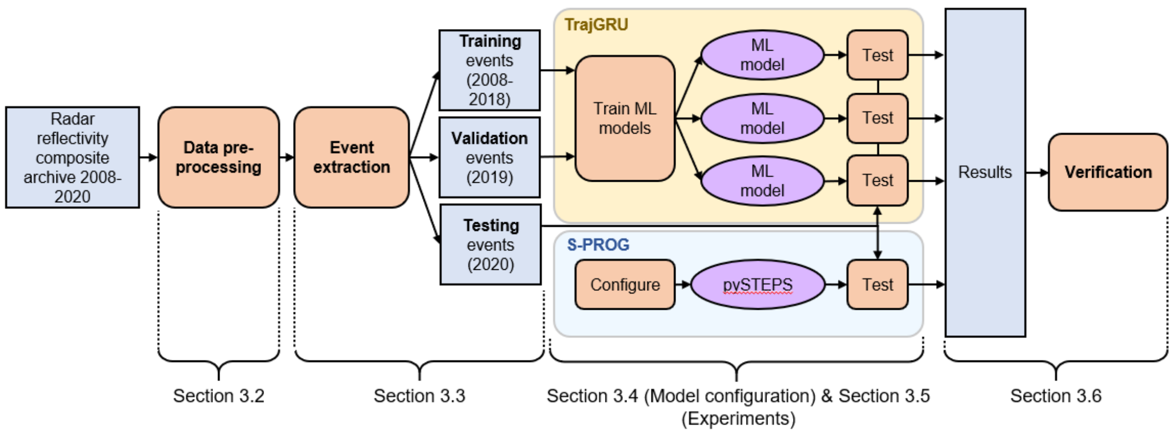


Figure 3.1: Schematisation of research process

In Figure 3.1 a schematisation of the research is presented. The radar reflectivity composites undergo pre-processing, as described in Section 3.2. Events are extracted from the radar archive, both heavy summer events for testing, and training and corresponding validation events for training the deep learning model. This process is described in Section 3.3. In Section 3.4 the model set-up is stated and Section 3.5 lists the experiments that are performed. In Section 3.6, verification methods that are used for model performance analysis are discussed.

3.2. Data pre-processing

The dataset that was used in this study is low level processed. The data has not been validated with ground observations and there is still clutter present in the images. Since the focus of this research is on heavy precipitation, removing clutter with high reflectivity values is more important than removing clutter with low reflectivity values. In order to visualize the nature of the high reflectivity values, density maps were created for several reflectivity intervals, see Figure 3.2. The majority of the values in the visualized intervals seem to be of non-meteorological nature: The stripes are caused by ship radars and correspond with the vessel density map in Figure 3.3, the dots in the upper left corner are expected to originate from off-shore wind farms. Because only a small fraction of reflectivity values above 50 dBZ seems to be of meteorological nature, the decision was made to put all these values to 0 dBZ.

The deep learning model that is used has a fixed input size of 480x480, which will be referred to as the model input. Model performance of both models was measured in a domain of 360x360 at the center

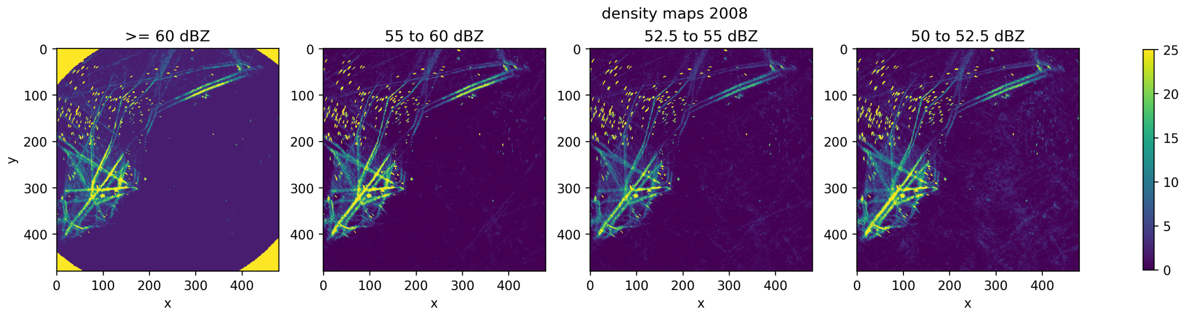


Figure 3.2: Density maps of pixels in reflectivity intervals for 2008.

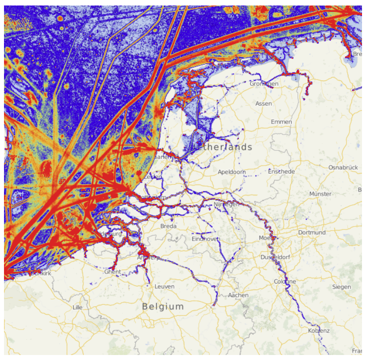


Figure 3.3: Vessel density map, retrieved from https://www.vesselfinder.com/

of the model domain, which will be referred to as the ’research domain’, see Figure 3.4.

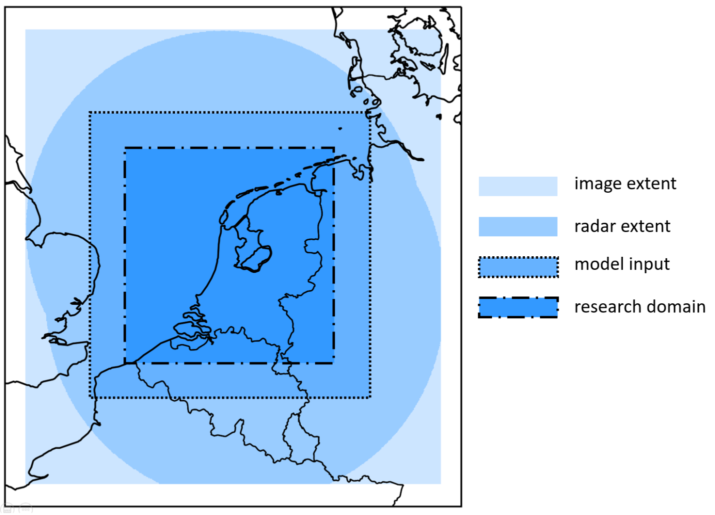


Figure 3.4: Image extent: the 700x765 array the data is provided in. Radar extent: the pixels containing radar measurements. Model input: 480x480 array. Research domain: 360x360 array.

3.3. Event extraction

The deep learning model that is used, takes 5 historical radar frames to predict 20 future frames (+100 minutes), which means sequences of 25 consecutive frames are needed for training, validation and testing. The goal of the research is to train the machine learning model to nowcast heavy, summer type precipitation. So the first step is to extract these type of events from the dataset. Then, an appropriate training dataset needs to be chosen to train the machine learning model on. The dataset is split into three parts, 2008-2018 (11 years) is used for training, 2019 for validation and 2020 for testing. In Figure 3.5, a schematic representation of the proposed sequence selection process is presented. First, all frames are assigned one of three labels: 'no rain', 'light rain' or 'heavy rain'. Then, events are extracted based on these labels. A sequence is defined as a series of 25 consecutive frame, and is categorized based on what labels are present within this series. There are two event categories: 'regular' and 'heavy', the latter being a subset of the former. Applying different conditions for labelling and sequencing results in different subsets of the data set that can be chosen from for training and testing purposes.

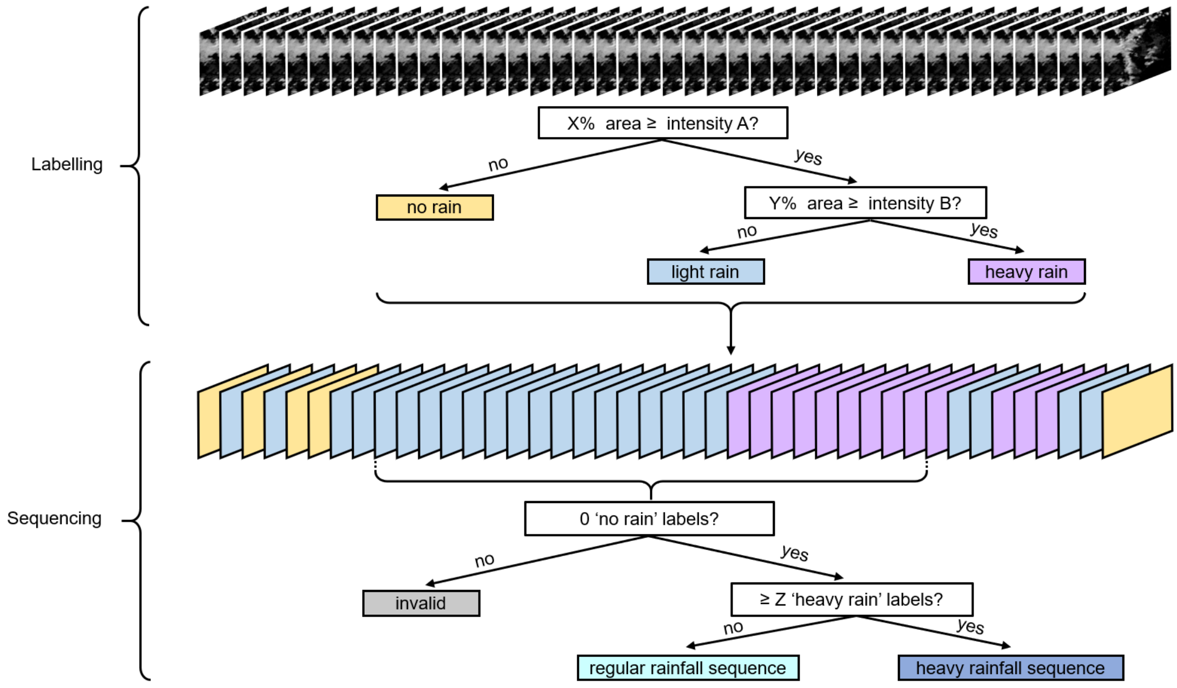


Figure 3.5: Schematic representation of sequence selection process. X and Y are area percentage thresholds, A en B are rainfall intensity thresholds in mm/h, Z is the minimum 'heavy rain' label count for a 'heavy rainfall sequence'.

3.3.1. Labelling

Labels are assigned based on spatial rainfall properties within the research domain. Two properties are defined for every frame: **wet area**, which is defined as the area of the domain where the rainfall intensity ≥ 1 mm/h, and **peak intensity**, the maximum rainfall intensity occurring in the domain. To prevent that the peak intensity is caused by a spurious pixel a minimum amount of 0.05% of the area (64 pixels) is required. Labels are then assigned based on whether the frames fulfill a certain set of conditions, for example a minimum wet area of 1% and a minimum peak intensity of 10 mm/h. With these conditions, a dry frame ('no rain') is characterized by a wet area < 1%. Light frames are characterized by a wet area ≥ 1% and a peak intensity < 10 mm/h. Heavy rainy frames are characterized by a wet area ≥ 1% and a peak intensity ≥ 10 mm/h. In order to investigate the balance between labelling conditions and resulting numbers of labels and sequences, six different labelling conditions were constructed. In Table 3.1, the conditions and the resulting percentages of labels in the training dataset (2008-2018) are presented.

3.3.2. Sequencing

Sequences are classified based on what labels are present in 25 consecutive frames. If there is any 'no rain' frame within the sequence, it is automatically deemed invalid. If all frames have either a 'light rain' or 'heavy rain' label, the sequence is classified as a 'regular rainfall sequence'. If a sequence also contains ≥ 12 frames with a 'heavy rain' label, the event is also classified as a 'heavy rainfall sequence'. See Table 3.2 for the resulting numbers of train and test sequences for the labelling conditions

Name	Condition for light rain	Peak intensity for heavy rain	% no rain	% light rain	% heavy rain
C1	≥ 1% at 1 mm/h	10 mm/h	77.80 %	15.17 %	7.02 %
C2	≥ 1% at 1 mm/h	20 mm/h	77.80 %	18.97 %	3.23 %
C2	≥ 1% at 1 mm/h	30 mm/h	77.80 %	20.49 %	1.70 %
C4	≥ 2% at 1 mm/h	10 mm/h	86.20 %	8.90 %	4.90 %
C5	≥ 2% at 1 mm/h	20 mm/h	86.20 %	11.95 %	1.85 %
C6	≥ 2% at 1 mm/h	30 mm/h	86.20 %	13.04 %	0.76 %

Table 3.1: Six different labelling condition with resulting percentages of assigned labels 2008-2018.

introduced in Table 3.1.

Name	Min area of ≥ 1 mm/h RR for 25 consecutive frames	Min peak RR for ≥ 12 of 25 frames	No. of train sequences	No. of test sequences
All_c1	1% at 1 mm/h	None	185,324	15,872
Heavy_c1	1% at 1 mm/h	10 mm/h	65,726	4,613
Heavy_c2	1% at 1 mm/h	20 mm/h	26,875	1,529
Heavy_c3	1% at 1 mm/h	30 mm/h	13,292	612
All_c4	2% at 1 mm/h	None	107,855	8,976
Heavy_c4	2% at 1 mm/h	10 mm/h	44,611	3,250
Heavy_c5	2% at 1 mm/h	20 mm/h	15,967	946
Heavy_c6	2% at 1 mm/h	30 mm/h	6,264	476

Table 3.2: Resulting number of train and test sequences for the different labelling conditions Name prefix "All_" denotes the 'regular rainfall sequences', prefix "Heavy_" refers to the 'heavy rainfall sequences'.

3.3.3. Selecting test sequences

Figure 3.6 shows how the different labelling conditions affect the amount and distribution of heavy rainfall sequences in the test year 2020. It is clearly visible that as the labelling conditions become more strict, the total number of events goes down and the events get more concentrated around the summer months, resulting in events only in June and August for the conditions which require a 30 mm/h peak intensity. Because heavy summer precipitation is the focus of this research, it is chosen to continue with the strictest condition because this results in the heaviest precipitation events. Thus, the test set is chosen to be heavy_c6 (bottom right in Figure 3.6). The result is 13 different rainfall events of different lengths as presented in Table 3.3. The distribution of pixel values within the sequences in the test events is presented in Figure 3.7.

Event no.	Start event	Stop event	Event duration	No. of test events
1	12 Jun 2020 18:40	12 Jun 2020 23:40	05:00	37
2	14 Jun 2020 12:20	14 Jun 2020 15:50	03:30	19
3	17 Jun 2020 14:35	17 Jun 2020 19:55	05:20	41
4	26 Jun 2020 18:00	26 Jun 2020 21:35	03:35	20
5	27 Jun 2020 06:25	27 Jun 2020 10:00	03:35	20
6	27 Jun 2020 12:10	27 Jun 2020 14:40	02:30	7
7	11 Aug 2020 18:35	11 Aug 2020 20:40	02:05	2
8	13 Aug 2020 15:25	13 Aug 2020 19:30	04:05	26
9	14 Aug 2020 01:00	14 Aug 2020 07:55	06:55	60
10	16 Aug 2020 13:20	16 Aug 2020 01:05	11:45	118
11	17 Aug 2020 11:15	17 Aug 2020 20:35	09:20	89
12	20 Aug 2020 20:45	20 Aug 2020 23:45	09:00	13
13	21 Aug 2020 01:55	21 Aug 2020 05:50	03:55	24

Table 3.3: Event properties of the events in the test set.

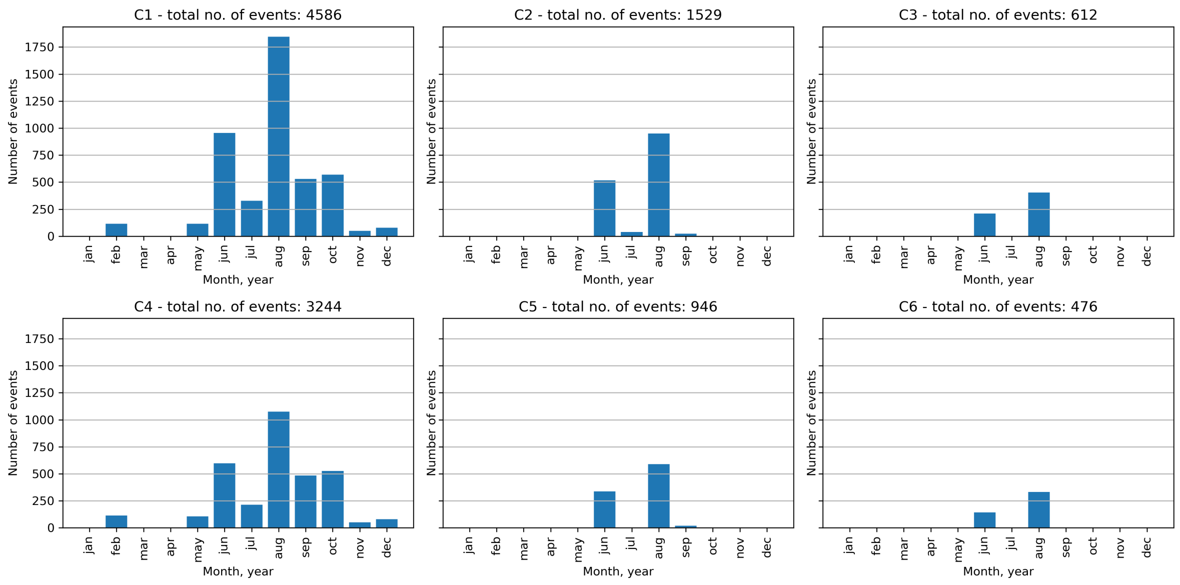


Figure 3.6: Temporal distribution of test events for different condition sets.

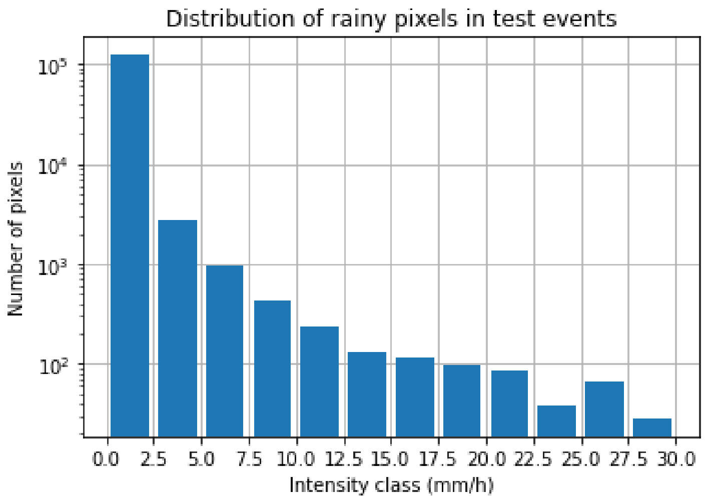


Figure 3.7: Bar chart of number of pixels in different intensity classes for all sequenes in the test set. Note the logarithmic scale: 96% of all pixels fall within the first category, 0 - 2.5 mm/h.

3.4. Model set-up

3.4.1. TrajGRU

The model is implemented using PyTorch and is trained on an 8 GB Nvidia GeForce RT2080 GPU. Batch size is set to 2. Any larger batch size exceeds the GPU's memory. Every training, 100,000 iterations are performed. This takes approximately 70-90 hours. A learning rate (LR) schedule with step-decay is used, with a decay factor of LR * 0.1 at the 30,000th and 60,000th iteration. Learning rate decay factors of 0.3 and 0.7 were also tried in some preliminary experiments. Both led to overfitting, so 0.1 is used for every model in this research. After pre-processing as described in Section 3.2, the radar reflectivity images are normalized with 0 = -32 dBZ and 1 = 50 dBZ.

3.4.2. S-PROG

Model set-up and parameters were adopted from Imhoff et al. (2020), which is the same set-up that KNMI uses for their STEPS implementation. The number of cascade levels is set to 8, rainfall is thresholded at 0.1 mm/h minimum, and the last 4 time steps (t, t-1, t-2, and t-3) are used to determine the motion field. The data pre-processing step of removing all values ≥ 50 dBZ is applied here too, but the input data is not cropped. The fixed input size is a limitation of the deep learning model and not of S-PROG. The results are however only evaluated in the research domain as defined in Figure 3.4.

3.5. Experiments

In order to answer research question 1, two concepts in terms of deep learning training set up are explored: the influence of the training data (Section 3.5.1) and the influence of the loss function (3.5.2). The resulting deep learning models that need to be trained for this are listed in Section 3.5.3. To answer research question 2, these models are compared to S-PROG.

3.5.1. Influence of training data

When constructing a training dataset for a deep learning model two things should be kept in mind:

1. The training data needs to contain enough information to learn the task at hand, in other words: the training data needs to be representative for the testing data

2. The training dataset needs to be sufficiently large to adequately train the model. Preferably as large as possible, because in theory, the performance of a deep learning model keeps increasing as the dataset size increases (e.g. Hestness et al., 2017).

Point 1 can be achieved by applying the same conditions that were used to extract the test sequences, to extract the train sequences. However this yields only 6,264 sequences from the training years. This is not enough to train a deep learning model, so a different subset needs to be chosen. A balance needs to be found between the amount of examples in the training dataset, and to what extend the sequences in the training dataset represent the sequences in the test dataset. Three models with three different training datasets are trained in order to find this balance. As the training dataset size increases, the share of heavy events decreases. The subsets that are tested are all_c1 (wet area ≥1% in all frames), all_c4 (wet area ≥2% in all frames) and heavy_c1 (wet area ≥1% in all frames + peak intensity of ≥10 mm/h in ≥12 frames), as defined in Table 3.2, listed from largest (smallest share of heavy events, thus less representative) to smallest (largest share of heavy events, thus more representative). These training datasets will be referred to as L, M and S, respectively. The amount and distribution of the events within these subsets is shown in Figure 3.8. It is visible that all datasets have more training sequences in summer than in other seasons of the year. This is most pronounced in dataset S, which is the only dataset that employed a minimum peak intensity.

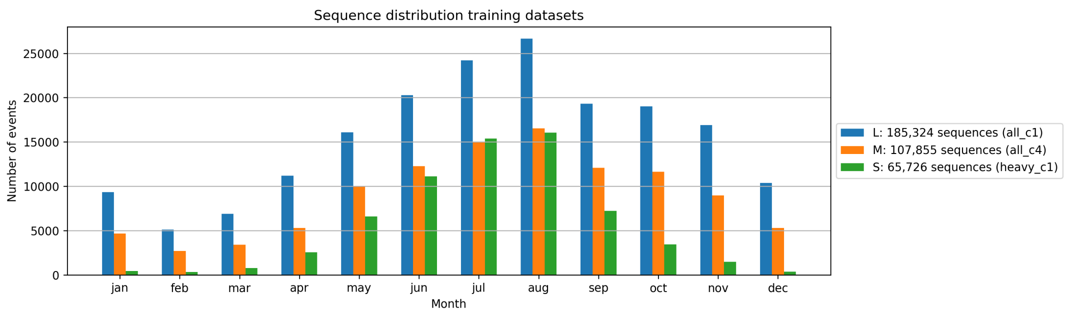


Figure 3.8: Number of sequences per month for the three different subsets of the training data.

3.5.2. Influence of loss function

Several loss functions are explored. A simple RMSE loss, a balanced loss function to emphasize high rainfall intensities and the application of a mask to exclude pixels at the border of the model domain.

RMSE loss The average root mean square error between the observation and the prediction is used as the loss function:

$$\text{Loss} = \text{RMSE} = \sqrt{\frac{1}{N * 360 * 360} \sum_{n=1}^{N} \sum_{i=1}^{360} \sum_{j=1}^{360} (F_{n,i,j} - O_{n,i,j})^2} \tag{3.1}$$

Balanced loss To emphasize the importance of predicting high values correctly, Shi et al. (2017) introduced a balanced loss function: a combination of mean squared error (MSE) and mean absolute error (MAE), where the weight of the error depends on observed rainfall intensity. Weights are determined by discrete rainfall intensity classes. In this research, a continuous scheme is used: the weight of the error of an estimated value is equal to the observed rainfall intensity in mm/h. The loss function is thus defined as following:

$$\text{Loss} = \text{B-MSE} + \text{B-MAE} \tag{3.2}$$

$$\text{B-MSE} = \frac{1}{N * 360 * 360} \sum_{n=1}^{N} \sum_{i=1}^{360} \sum_{j=1}^{360} w_{n,i,j} (F_{n,i,j} - O_{n,i,j})^2 \tag{3.3}$$

$$\text{B-MAE} = \frac{1}{N * 360 * 360} \sum_{n=1}^{N} \sum_{i=1}^{360} \sum_{j=1}^{360} w_{n,i,j} |F_{n,i,j} - O_{n,i,j}| \tag{3.4}$$

$$w_{n,i,j} = \begin{cases} 1 & \text{if } R_{n,i,j} \leq 1 \\ R_{n,i,j} & \text{if } 1 < R_{n,i,j} \leq 30 \\ 30 & \text{if } R_{n,i,j} > 30 \end{cases} \tag{3.5}$$

Where N is the number of output frames, and $F_{n,i,j}$, $O_{n,i,j}$ and $w_{n,i,j}$ are the forecast, observation and weight, respectively, corresponding to the (i,j)th pixel in the nth frame. The weight $w_{n,i,j}$ is equal to the rainfall intensity $R_{n,i,j}$ observed at this point, with a minimum weight of 1 and a maximum weight of 30. The rainfall intensity is derived by transforming pixel value to dBZ and using (2.1) to transform to rainfall intensity in mm/h.

Border mask Because the model cannot be expected to predict what is coming in from outside the model domain, it is decided not to penalize errors made within close range of the border of the domain. The exclusion of these pixels is done by applying a mask within the loss function, that multiplies the pixels that are excluded from evaluation by 0 and multiplies the other pixels by 1. It was chosen to exclude a range of 60 km from the border of the model domain, which corresponds to the research domain defined in 3.4.

3.5.3. List of models

In order to determine the influence of the above explained concepts, five different deep learning models are trained, see Table 3.4. Models 1, 2 and 3 are compared in the first experiment, where the influence of the training dataset is explored. Models 3, 4 and 5 are compared in the second experiment, where the influence of the loss function is explored. By compare models 3 and 4 the effect of the mask can be distinguished and by models 3 and 5 the effect of the balanced loss function can be analysed. In experiment three, the deep learning models are compared to S-PROG.

3.6. Verification

3.6.1. Continuous metrics

Mean Absolute Error MAE measures the average magnitude of the absolute error.

$$MAE = \frac{1}{360 * 360} \sum_{i=1}^{360} \sum_{j=1}^{360} |F_{i,j} - O_{i,j}| \tag{3.6}$$

	Training data	Loss Function
1	Small: 65,726 samples (heavy_c1)	B-MSE + B-MAE + mask
2	Medium: 107,855 samples (all_c4)	B-MSE + B-MAE + mask
3	Large: 185,324 samples (all_c1)	B-MSE + B-MAE + mask
4	Large: 185,324 samples (all_c1)	B-MSE + B-MAE
5	Large: 185,324 samples (all_c1)	RMSE + mask

Table 3.4: List of performed ML experiments

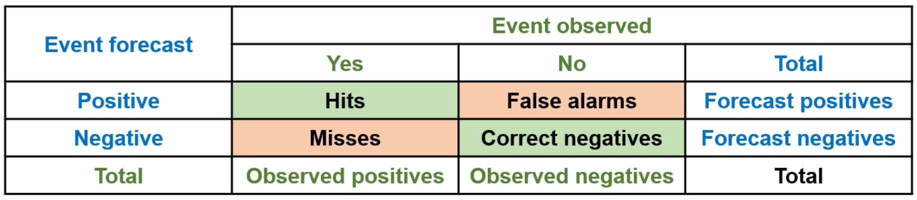
Root mean square error RMSE measures the square root average of the squared error. This puts more emphasis on the larger errors and less emphasis on smaller errors.

$$RMSE = \sqrt{\frac{1}{360 * 360}\sum_{i=1}^{360}\sum_{j=1}^{360}(F_{i,j} - O_{i,j})^2} \tag{3.7}$$

Conditional absolute error and bias As shown in Figure 3.7, 96% of all pixels in the test sequences are between 0 - 2.5 mm/h. This means that any averaged error is dominated by errors made in the smallest rainfall intensities. In order to also be able to say something about errors made at different rainfall intensities, (mean) Absolute error and (mean) bias (sum of the errors) are calculated separately for different rainfall intensity classes. For example: The total absolute error for intensity class 0 - 2.5 mm/h represents the sum of the absolute error for the pixels where in the observations 0 - 2.5 mm/h rainfall rate was measured. The mean values are the error values averaged over the area within this intensity class.

3.6.2. Categorical metrics

To calculate categorical metrics both the observation and forecast maps have to be converted into dichotomous maps (positive/negative) based on a chosen threshold. If a pixel value is smaller than threshold, the value is 'negative', and if a pixel value larger than or equal to the threshold, the value is 'positive'. By comparing the maps, a contingency table as in figure 3.9 can be constructed. The categorical metrics are then calculated from the number of hits (true positives), misses (false negatives), false alarms (false positives) and correct negatives.



Event forecast	Event observed		
	Yes	No	Total
Positive	Hits	False alarms	Forecast positives
Negative	Misses	Correct negatives	Forecast negatives
Total	Observed positives	Observed negatives	Total

Figure 3.9: Contingency table

Probability Of Detection POD expresses what ratio of observed positives was predicted correctly. It ranges form 0 to 1, where 1 is the perfect score, which means all observed positives were predicted. Score ignores false alarms, so it is sensitive to over-forecasting: score increases when more positives are forecasted.

$$POD = \frac{\text{hits}}{\text{hits} + \text{misses}} \tag{3.8}$$

False Alarm Ratio FAR expresses what ratio of forecast positives were false alarms. It ranges from 0 to 1, where 0 is the perfect score, which means that none of the predicted positives were false alarms.

$$FAR = \frac{\text{false alarms}}{\text{hits} + \text{false alarms}} \tag{3.9}$$

Critical Success Index CSI measures how well the forecasted positives correspond to observed positives. It is similar to the POD, but it also penalizes false alarms. It ranges from 0 to 1, where 0 indicates no skill and 1 is the perfect score and indicates a perfect forecast.

$$CSI = \frac{\text{hits}}{\text{hits} + \text{misses} + \text{false alarms}} \tag{3.10}$$

Frequency bias measures the ratio between forecasted positives and observed positives. Ranges from 0 to ∞. A score >1 indicates overforecasting and a score <1 indicates underforecasting. 1 is the perfect score: the number of forecasted positives is the same as the number of observed positives, but this does not say anything about whether these forecasted positives were correct.

$$BIAS = \frac{\text{forecasted positives}}{\text{observed positives}} = \frac{\text{hits} + \text{false alarms}}{\text{hits} + \text{misses}} \tag{3.11}$$

Fractions Skill Score FSS is a spatial verification metric that scores model performance on different intensity and different spatial scales. It ranges from 0 to 1, where 0 indicates no skill and 1 indicates a perfect forecast. A forecast is deemed useful if FSS > FSS$_{useful} = 0.5 + \frac{f_0}{2}$, where f_0 is the area of the observed scale divided by the area of the observed domain. With this FSS$_{useful}$, the smallest scale for a skillful forecast can be determined per intensity threshold, or, the maximum skillful lead time can be determined for a desired scale and intensity. For a deterministic forecast with threshold n the FSS is defined as in equation 3.12. (Roberts & Lean, 2008)

$$FSS_{(n)} = 1 - \frac{MSE_{(n)}}{MSE_{(n)ref}} \tag{3.12}$$

where

$$MSE_{(n)} = \frac{1}{N_x N_y} \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} [O_{(n)i,j} - M_{(n)i,j}]^2 \tag{3.13}$$

$$MSE_{(n)ref} = \frac{1}{N_x N_y} \left[\sum_{i=1}^{N_x} \sum_{j=1}^{N_y} O_{(n)i,j}^2 + \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} M_{(n)i,j}^2 \right] \tag{3.14}$$

where N_x and N_y are the number of columns and rows of n x n cells, and $O_{(n)i,j}$ and $M_{(n)i,j}$ are the fraction of pixels exceeding the set threshold within cell i,j in the observation and the forecast, respectively. $MSE_{(n)}$ is then the mean squared error of these fractions, and $MSE_{(n)ref}$ represents the largest possible MSE between the observation and the forecast. See Figure 3.10 for a visualization and example calculation of the FSS for one cell.

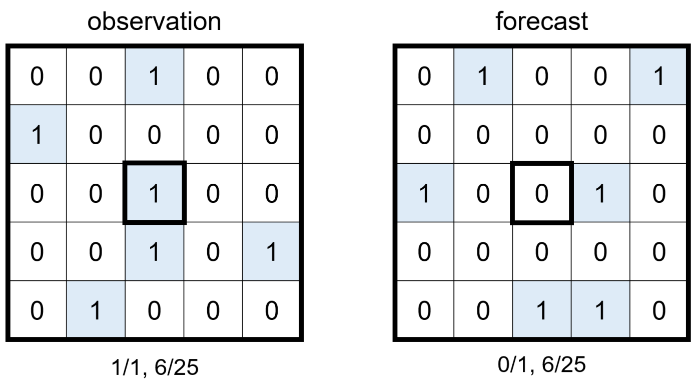


Figure 3.10: FSS at scale 1: $O_{(1)}$ = 1/1, $F_{(1)}$ = 0/1, so $MSE_{(1)}$ = 1 and $MSE_{(1)ref}$ = 1. This yields $FSS_{(1)}$ = 0. FSS at scale 5: $O_{(5)}$ = 6/25, $F_{(1)}$ = 6/25, so $MSE_{(1)}$ = 0 and $MSE_{(1)ref}$ = 36/625. This yields $FSS_{(5)}$ = 1. Adapted from Roberts & Lean (2008).

4 Results

In this chapter the model performance of all experiments are presented. In Section 4.1 the different TrajGRU training set-ups are compared in two parts: in Section 4.1.1 the different training datasets are compared and in section 4.1.2 the different loss functions. In Section 4.2, model performance between the TrajGRU models and S-PROG are compared. For every experiment, the following things are presented: a visualization of a +100 minute prediction of one test event, a graph with MAE and RMSE, a table listing CSI, POD, FAR and frequency bias for 1, 5, 10 and 20 mm/h thresholds and figures showing FSS-based maximum skillful lead times for several spatial spatial scales and intensity thresholds.

4.1. Influence of training data

In the first experiment, the influence of the training dataset is explored. Three different training datasets are explored with varying dataset size to heavy events ratios. The different datasets are referred to as TrajGRU S, TrajGRU M and TrajGRU L, listed from smallest (largest share of heavy events) to largest (smallest share of heavy events). Figure 4.2 shows that TrajGRU L consistently performs better in terms of MAE and RMSE. TrajGRU S yields the largest errors. In Figure 4.1 it seems that with decreasing dataset size and increasing share of heavy events, more high rainfall intensities are predicted. The values reported in Table 4.1 confirm this behaviour with the higher observed frequency bias, especially for 10 and 20 mm/h rainfall intensities. This results in high rainfall intensities being detected more often, shown by an increased POD. However high value are also predicted wrongly more often (false alarms), resulting in an increased FAR. It should be note that the differences mentioned are very small. In terms of the maximum skillful lead times noted in Figure 4.3 and the absolute error and bias calculated for different intensity classes presented in Figure 4.4, the differences between the three models are very small.

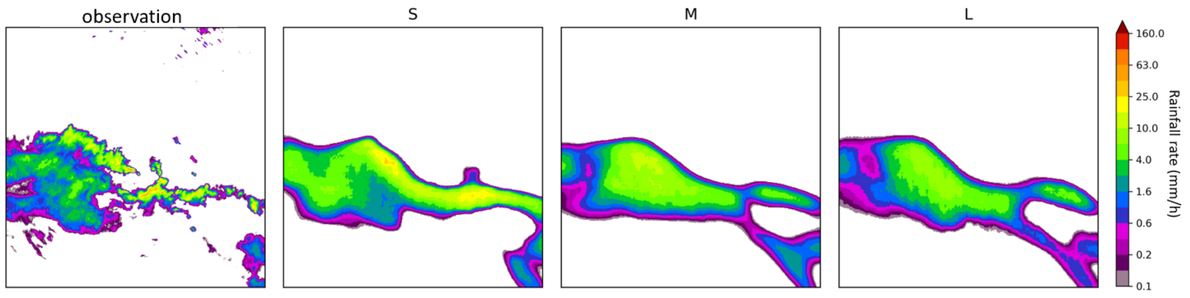


Figure 4.1: +100 minute predictions for t_0 = 16-08-2020 17:20, for the TrajGRU models with different training datasets.

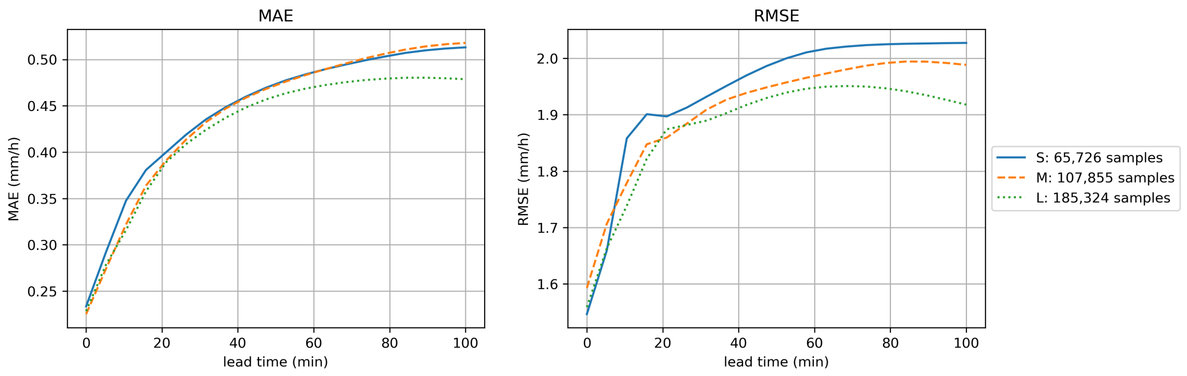


Figure 4.2: Mean absolute error and root mean square error for TrajGRU with different training data compositions.

	CSI				POD			
	≥1 mm/h	≥5 mm/h	≥10 mm/h	≥20 mm/h	≥1 mm/h	≥5 mm/h	≥10 mm/h	≥20 mm/h
S	0.474	0.218	0.129	0.041	0.736	0.474	0.258	0.070
M	0.476	0.218	0.121	0.036	0.743	0.469	0.234	0.060
L	0.474	0.215	0.115	0.031	0.729	0.444	0.214	0.051

	FAR				Frequency bias			
	≥1 mm/h	≥5 mm/h	≥10 mm/h	≥20 mm/h	≥1 mm/h	≥5 mm/h	≥10 mm/h	≥20 mm/h
S	0.438	0.704	0.681	0.494	1.384	1.736	1.127	0.416
M	0.438	0.701	0.662	0.388	1.390	1.742	1.011	0.362
L	0.434	0.701	0.675	0.409	1.350	1.572	0.916	0.280

Table 4.1: CSI, POD, FAR and Frequency bias for TrajGRU with different training datasets.

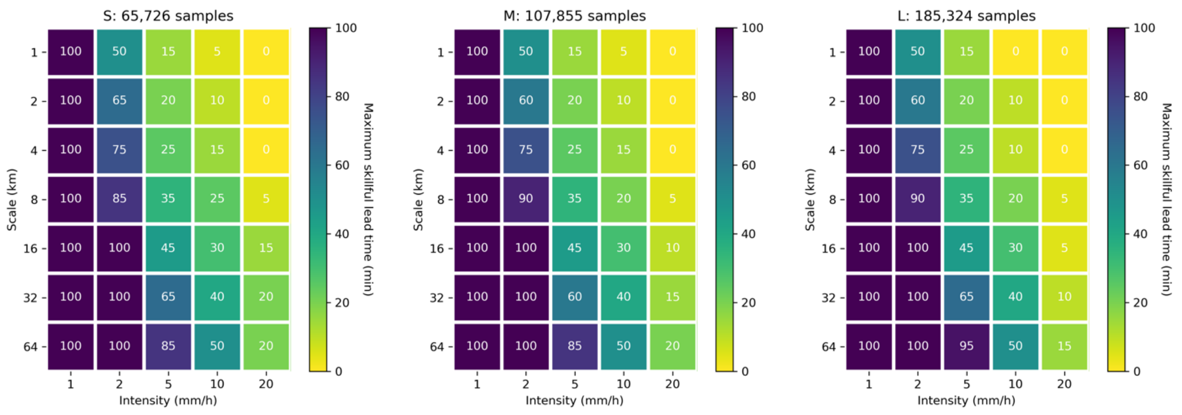


Figure 4.3: Maximum skillful lead times based on FSS for different scales and rainfall intensities for TrajGRU with different training data compositions.

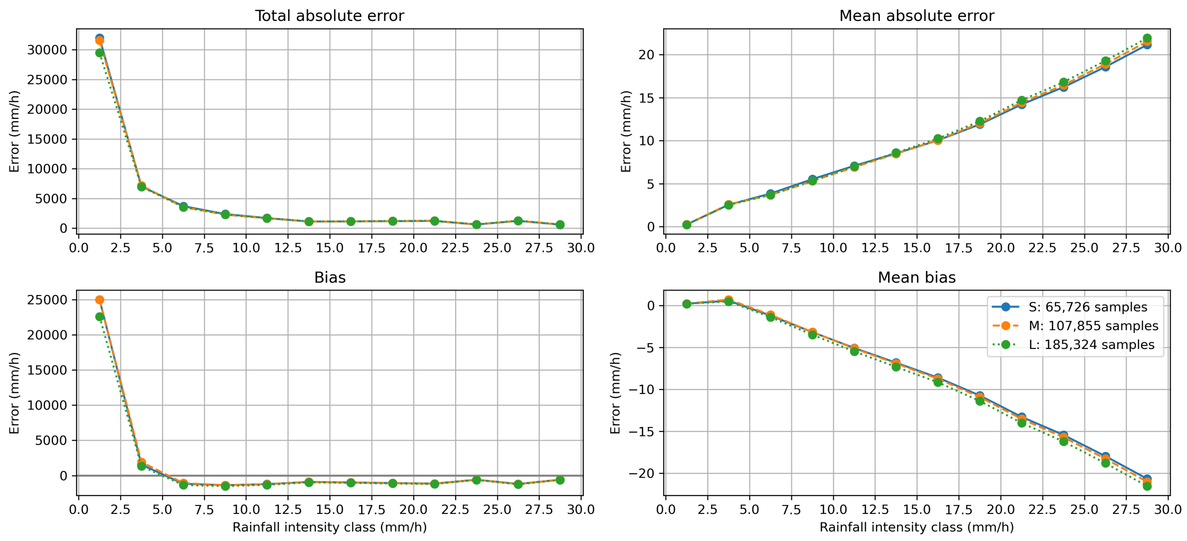


Figure 4.4: Absolute error and bias calculated separately in different rainfall intensity classes, averaged over all lead times, for TrajGRU with training datasets.

4.2. Influence of loss function

By comparing the model with loss function B-MSE + B-MAE + mask and the model with loss function RMSE + mask, the effect of the balanced loss function can be observed. By comparing the model with loss function B-MSE + B-MAE + mask and the model with loss function B-MSE + B-MAE, the effect of the mask can be distinguished.

4.2.1. Balanced loss function vs. RMSE

Figure 4.6 shows that in terms of MAE and RMSE, the model with the RMSE loss function consistently outperforms the balanced loss functions. In Figure 4.5 it is clear that the model with RMSE loss function predicts considerably smaller values than the models with the balanced loss functions. This behaviour is consistent with the metrics presented in Table 4.2: the balanced model shows a higher frequency bias, POD and CSI, but also a higher false alarm ratio. This is also confirmed by conditional absolute error and bias presented in Figure 4.9: the RMSE model consistently underestimates in all intensity classes, while the balanced model overestimates in the two lowest intensity classes, and underestimates to a lesser extent in the higher intensity classes. This is a direct result of how the balanced loss function is defined: in the lower intensity class, the weight of the error is the lowest so predicting correctly in this class was deemed less important in the training process. For the higher intensity classes, the weights of the errors are higher so predicting these values is deemed more important. Because of this, estimating higher intensities is encouraged because the penalty is small when it is wrong, and the reward is high when it's right.

Figure 4.7 and 4.8 show that in terms of maximum skillful lead times, the balanced loss function significantly outperforms the RMSE loss function at almost all scales and intensities.

4.2.2. Border mask

Concerning the effect of the application of the mask, Figure 4.6 shows that the model trained with the mask outperforms the model trained without the mask. The difference is smaller for the shorter lead times, and increases for larger lead times. The differences in terms of other metrics is small. The model with mask has slightly higher CSI, POD and FSS based skillful lead times, but also a slightly higher FAR.

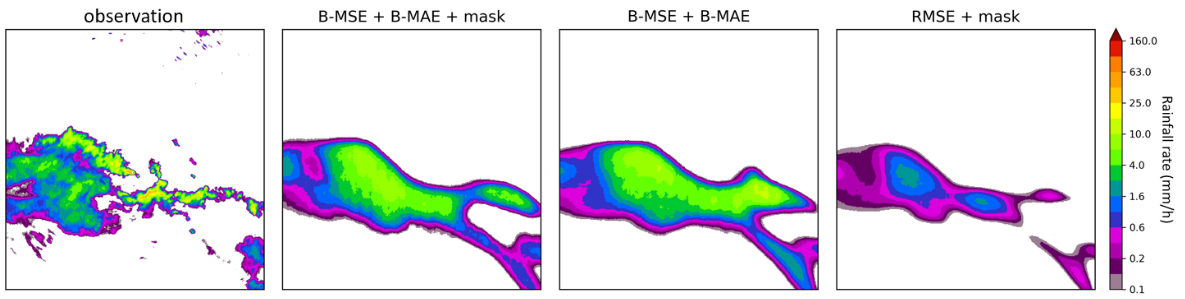


Figure 4.5: +100 minute predictions for t_0 = 16-08-2020 17:20, for the TrajGRU models with different loss functions.

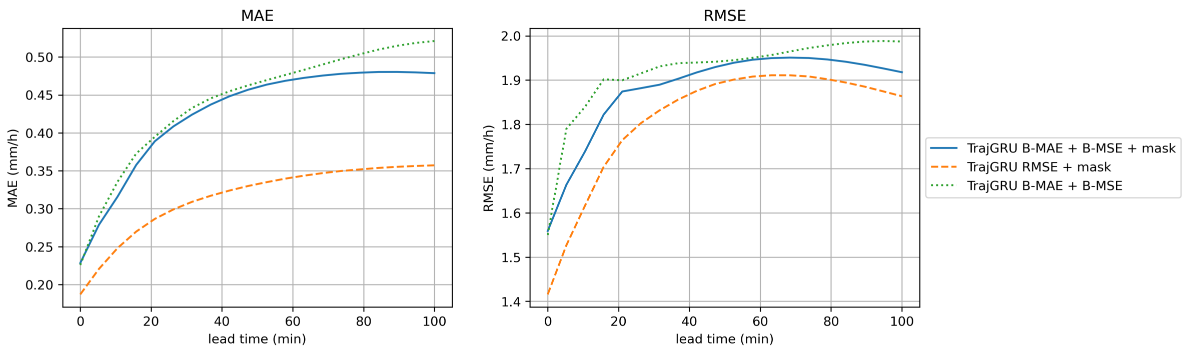
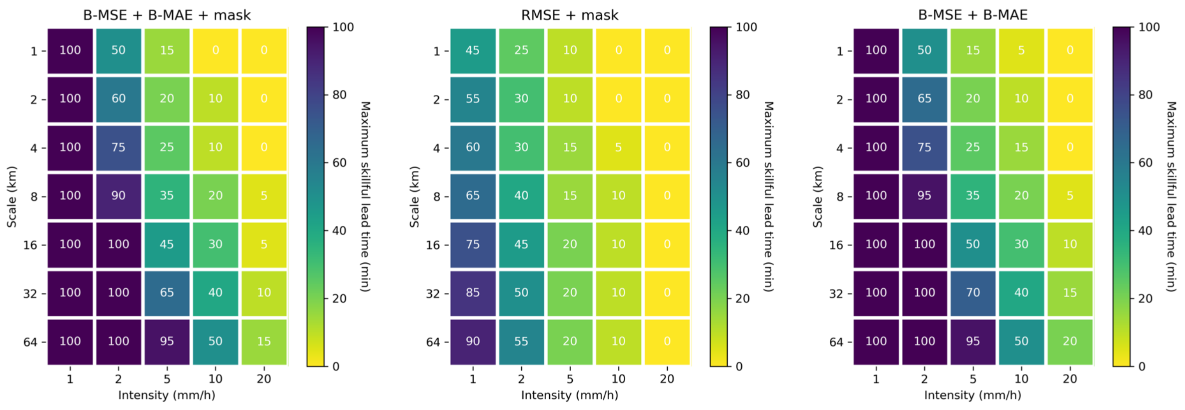


Figure 4.6: Mean absolute error and root mean square error for TrajGRU with different loss functions.

	CSI				POD			
	≥1 mm/h	≥5 mm/h	≥10 mm/h	≥20 mm/h	≥1 mm/h	≥5 mm/h	≥10 mm/h	≥20 mm/h
B+m	0.474	0.215	0.115	0.031	0.729	0.444	0.214	0.051
R+m	0.376	0.083	0.030	0.006	0.435	0.097	0.034	0.007
B	0.478	0.222	0.124	0.035	0.751	0.478	0.240	0.063

	FAR				Frequency bias			
	≥1 mm/h	≥5 mm/h	≥10 mm/h	≥20 mm/h	≥1 mm/h	≥5 mm/h	≥10 mm/h	≥20 mm/h
B+m	0.434	0.701	0.675	0.409	1.350	1.572	0.916	0.280
R+m	0.278	0.280	0.146	0.089	0.583	0.163	0.062	0.019
B	0.439	0.708	0.689	0.416	1.421	1.752	1.042	0.458

Table 4.2: CSI, POD, FAR and Frequency bias for TrajGRU with different loss functions: B-MSE + B-MAE + mask (B+m), RMSE + mask (R+m) and B-MSE + B-MAE (B).



Scale (km) \ Intensity (mm/h)	1	2	5	10	20
B-MSE + B-MAE + mask: 1	100	50	15	0	0
2	100	60	20	10	0
4	100	75	25	10	0
8	100	90	35	20	5
16	100	100	45	30	5
32	100	100	65	40	10
64	100	100	95	50	15
RMSE + mask: 1	45	25	10	0	0
2	55	30	10	0	0
4	60	30	15	5	0
8	65	40	15	10	0
16	75	45	20	10	0
32	85	50	20	10	0
64	90	55	20	10	0
B-MSE + B-MAE: 1	100	50	15	5	0
2	100	65	20	10	0
4	100	75	25	15	0
8	100	95	35	20	5
16	100	100	50	30	10
32	100	100	70	40	15
64	100	100	95	50	20

Figure 4.7: Maximum skillful lead times based on FSS for different scales and rainfall intensities for three different loss function configurations.

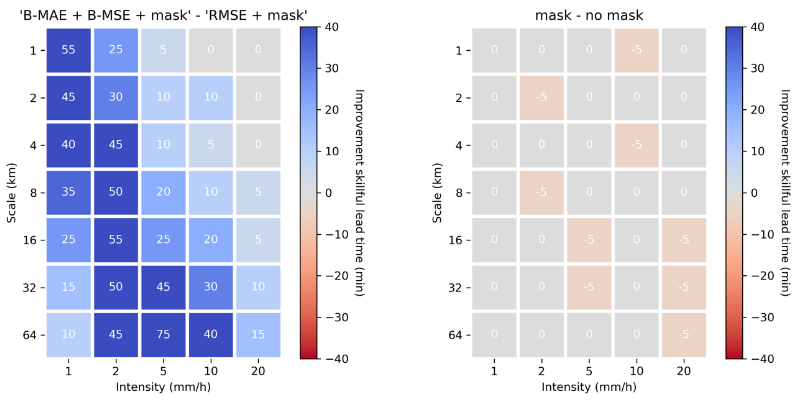


Figure 4.8: Improvement of skillful lead times when comparing (left) balanced loss function vs. RMSE loss functions and (right) mask vs. no mask.

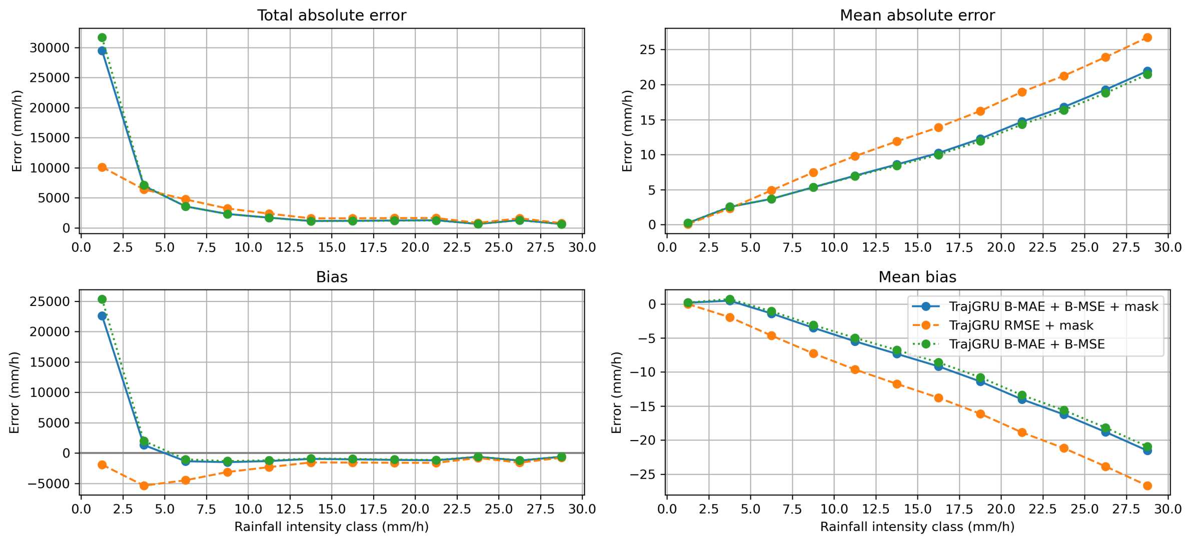


Figure 4.9: Absolute error and bias calculated separately in different rainfall intensity classes, averaged over all lead times, for TrajGRU with different loss functions.

4.3. Comparison to benchmark

Model performance of the benchmark S-PROG is compared to two versions of TrajGRU: the model with the RMSE loss, which resulted in low errors, but also low skill, and the model with the Balanced loss, which resulted in higher errors, but also higher skill. These models will referred to as 'TrajGRU RMSE' and 'TrajGRU Balanced', respectively.

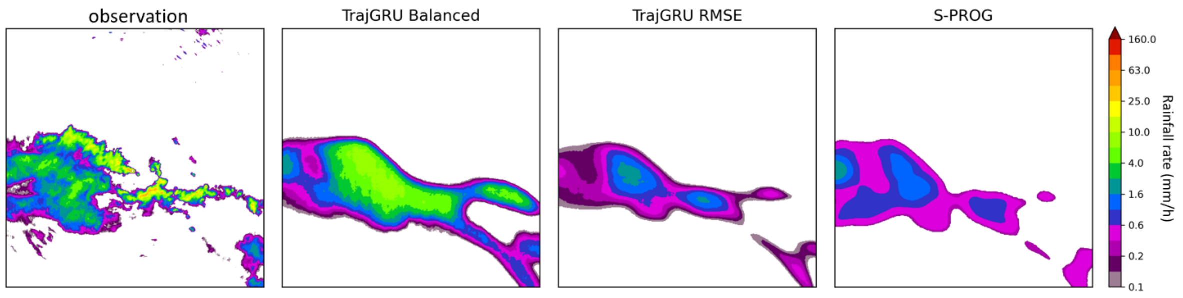


Figure 4.10: +100 minute predictions for t_0 = 16-08-2020 17:20, for TrajGRU Balanced, TrajGRU RMSE and S-PROG.

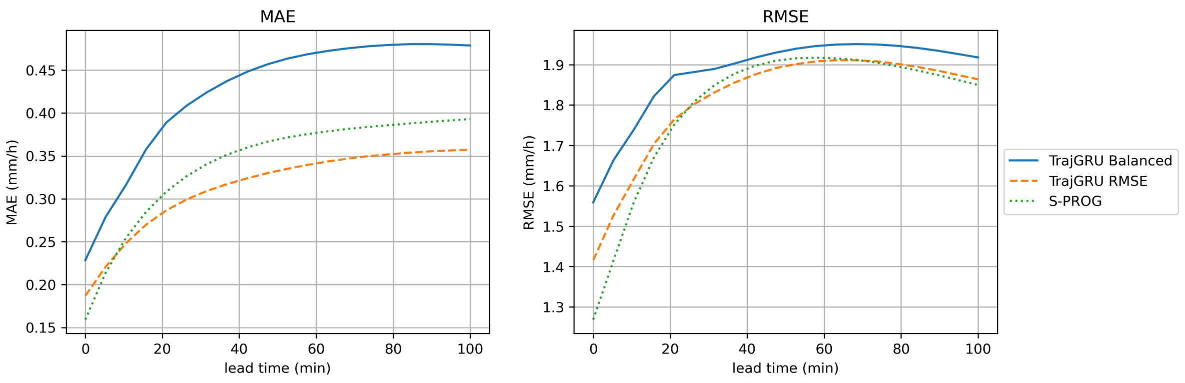


Figure 4.11: Mean absolute error and root mean square error for TrajGRU Balanced, TrajGRU RMSE and S-PROG.

	CSI				POD			
	≥1 mm/h	≥5 mm/h	≥10 mm/h	≥20 mm/h	≥1 mm/h	≥5 mm/h	≥10 mm/h	≥20 mm/h
B	0.474	0.215	0.115	0.031	0.729	0.444	0.214	0.051
R	0.376	0.083	0.030	0.006	0.435	0.097	0.034	0.007
S	0.400	0.121	0.059	0.026	0.507	0.162	0.077	0.034

	FAR				Frequency bias			
	≥1 mm/h	≥5 mm/h	≥10 mm/h	≥20 mm/h	≥1 mm/h	≥5 mm/h	≥10 mm/h	≥20 mm/h
B	0.434	0.701	0.675	0.409	1.350	1.572	0.916	0.280
R	0.278	0.280	0.146	0.089	0.583	0.163	0.062	0.019
S	0.368	0.507	0.332	0.144	0.816	0.418	0.208	0.084

Table 4.3: CSI, POD, FAR and Frequency bias for TrajGRU Balanced (B), TrajGRU RMSE (R) and S-PROG (S)

As can be seen in Figure 4.10, S-PROG creates smoothed predictions, similar to TrajGRU RMSE. In terms of MAE, TrajGRU RMSE outperforms S-PROG at all lead times except for the first 10-20 minutes. The difference increases with lead time. In terms of RMSE, S-PROG outperforms TrajGRU RMSE at the earliest and latest lead times, and TrajGRU RMSE is slightly better inbetween. TrajGRU Balanced consistently has the highest errors, both in terms of RMSE and MAE.

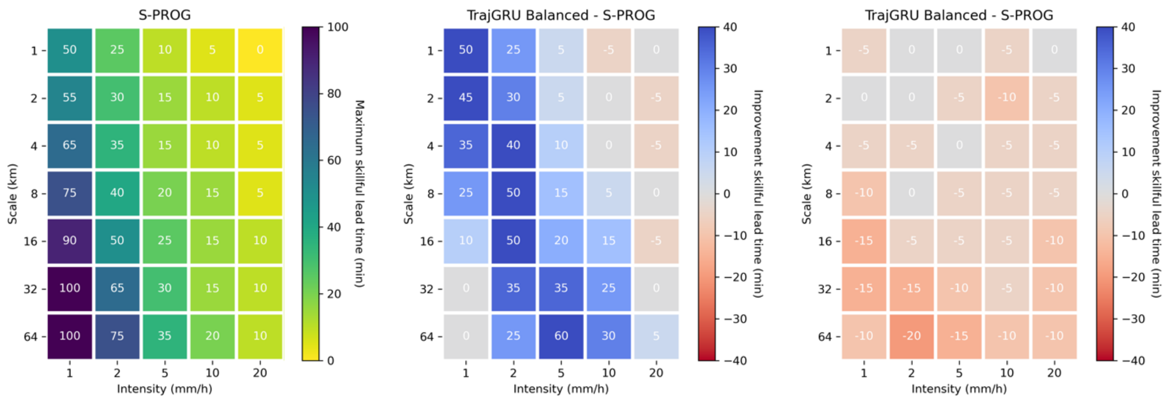


Figure 4.12: (left) Maximum skillful lead times based on FSS for S-PROG; (middle) Difference in skillful lead times between TrajGRU Balanced and S-PROG; (right) Difference in skillful lead times between TrajGRU RMSE and S-PROG.

Conversely, in terms of maximum skillful lead times as visible in Figure 4.12, TrajGRU Balanced performs the best, followed by S-PROG and then by TrajGRU RMSE. This is consistent with what has been observed in the previous experiments: Models with better performance at low rainfall intensities (see Figure 4.13) score better in terms of MAE and RMSE, but score lower in terms of CSI and POD (see Table 4.3) and FSS based skillful lead times (see Figure 4.12). Models with better performance at high rainfall intensities score worse in terms of MAE and RMSE, but higher in terms of CSI, POD and skillful lead times.

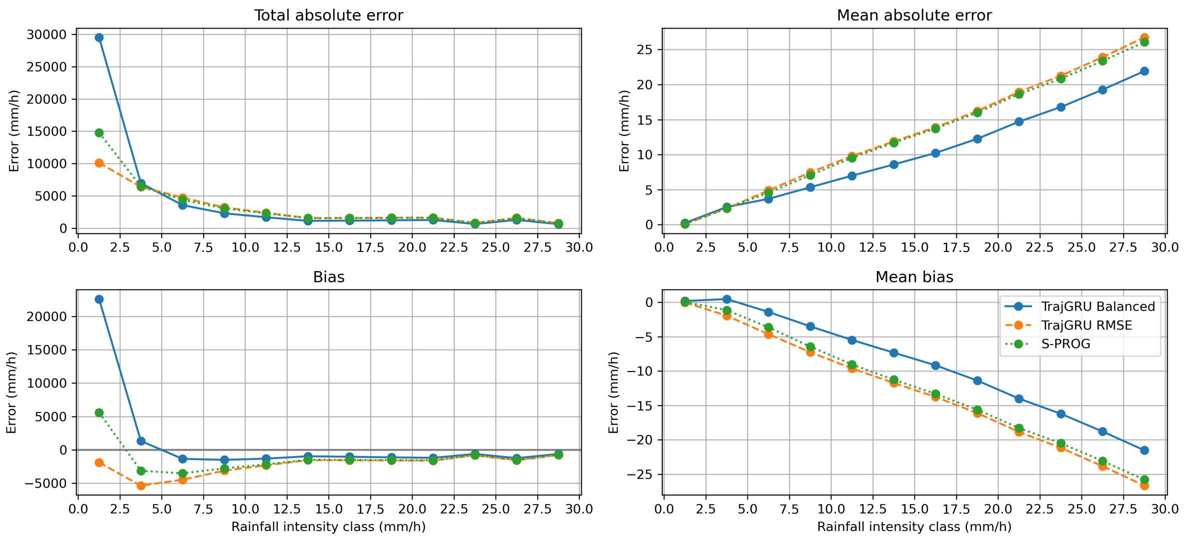


Figure 4.13: Absolute error and bias calculated separately in different rainfall intensity classes, averaged over all lead times, for TrajGRU Balanced, TrajGRU RMSE and S-PROG.

4.4. Case-by-case analysis

In the previous sections, average performance over all 476 sequences in the test set is analyzed. In this section, model performance is analyzed from an event point of view. The 476 test sequences are all part of the 13 events earlier presented in Table 3.3. In Figure 4.14 the RMSEs and difference in RMSE between TrajGRU Balanced and S-PROG are presented. Both models tend to follow approximately the same trends for the different events, and S-PROG shows a lower RMSE for almost all of the events.

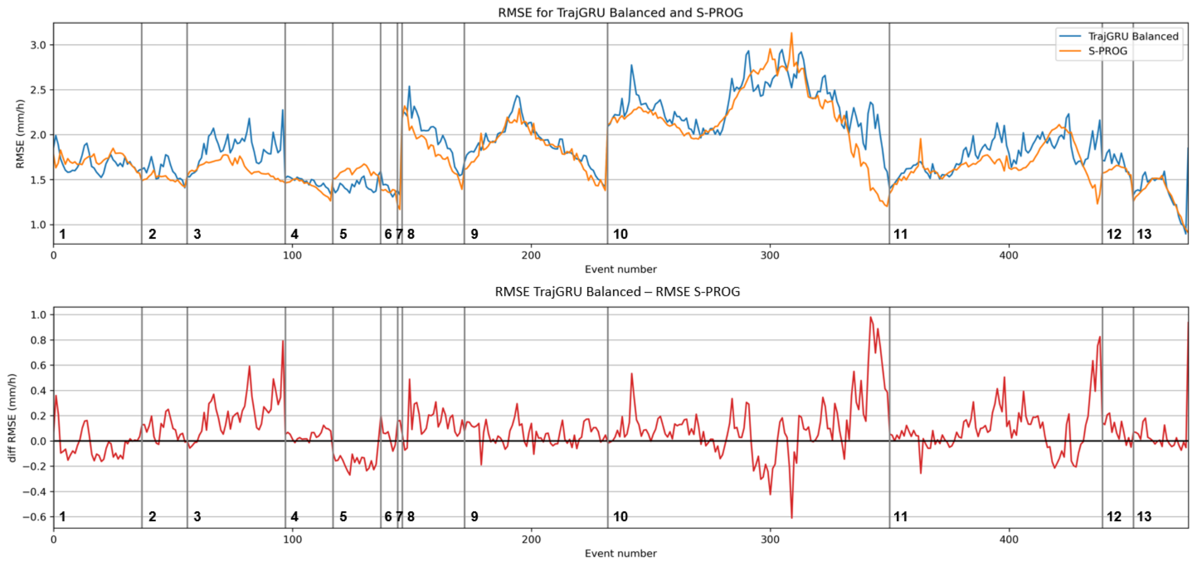


Figure 4.14: RMSE averaged over all lead times for the events as defined in Table 3.3. Events are separated by the grey vertical lines, event numbers are in the figure. Top: RMSE for both S-PROG and TrajGRU balanced. Bottom: RMSE TrajGRU - RMSE S-PROG.

To get better insight into the difference in model behaviour, an in depth analysis of two test sequences is provided: One where S-PROG significantly outperforms TrajGRU Balanced in terms of RMSE (see Figure 4.15) and one where TrajGRU Balanced significantly outperforms S-PROG (see Figure 4.16. The example where S-PROG outperforms TrajGRU Balanced shows that both models do capture the correct motion of the rainfall field, however the models predict different evolutions of the rainfall field. S-PROG correctly predicts dissipation, while TrajGRU Balanced predicts an increased area with high rainfall intensities, resulting in large errors. In the example where TrajGRU Balanced outperforms S-PROG the same difference between the models can be observed, however this time S-PROG incorrectly predicts dissipation, while TrajGRU Balanced correctly predicts an increase in area with high precipitation values at the right side of the rainfall field.

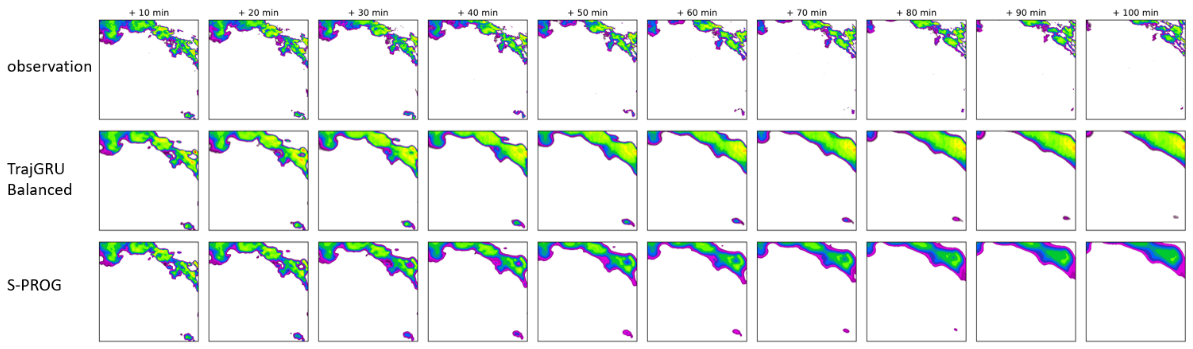


Figure 4.15: S-PROG outperforming TrajGRU Balanced. t_0 = 2020-08-16 22:50:00. Event no. 342 in Figure 4.14.

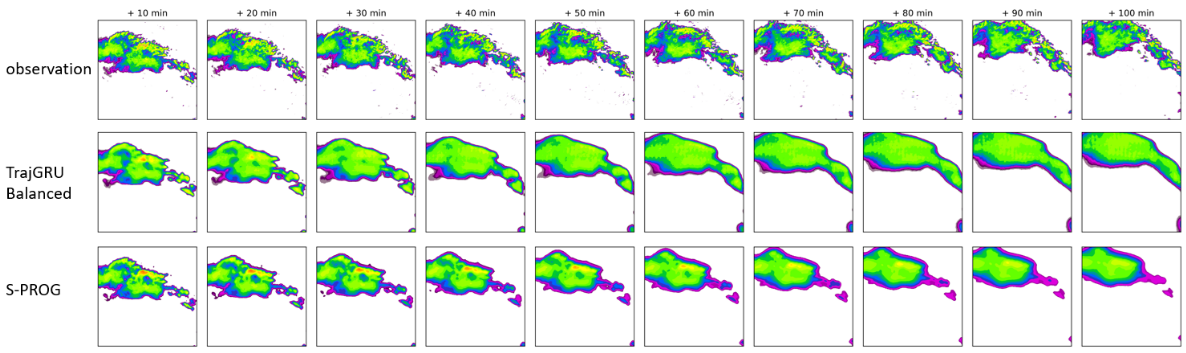


Figure 4.16: TrajGRU Balanced outperforming S-PROG. t_0 = 2020-08-16 20:05:00. Event no. 309 in Figure 4.14

5 Discussion

In this chapter the results presented in the previous chapter are discussed. The aim of this research was to explore how the choices made in the training set up of a deep learning model influence its ability to accurately nowcast heavy precipitation events. Different training datasets and different loss functions were applied. In Sections 5.1 and 5.2 the results of these experiments are discussed, respectively. The deep learning approach was also compared to a extrapolation-based benchmark, which is discussed in Section 5.3. Three additional topics, definition of model performance, data pre-processing and the predictability of events are discussed as well.

5.1. Influence of training data on model performance

As stated in the Methodology there are two important things to keep in mind when constructing a training dataset for a deep learning model: the data should be representative for the intended task, and the dataset should be large enough to adequately train the model. Because the same conditions that yielded the heavy summer precipitation events that were used for testing did not yield enough training events, less strict conditions were needed to create a large enough training dataset. Three different training datasets of different sizes were constructed: the larger the dataset, the smaller the share of heavy events, the less representative the data is. It was found that the largest training dataset scored the best in terms of RMSE and MAE. Results in terms of the other presented metrics showed only small differences. This means that between the tested training datasets, dataset size is more important than to what extent the data is representative.

In theory, the performance of a deep learning model keeps increasing as the dataset size increases (Hestness et al., 2017). In the case of the datasets tested in this research, increased dataset size resulted in less representative examples. Collecting more radar data can provide the opportunity to increase dataset size while also maintaining a certain level of representation of heavy precipitation events. However, there is not more radar data from the same dataset used in this research available: the complete archive of the product was used and any older data is at a different spatial scale (2.4 x 2.4 km instead of 1 x 1 km, see (Overeem et al., 2009)). A solution to this problem could be the use of transfer learning, where the size of the training dataset is increased by including radar data from other locations with similar rainfall regimes (e.g. Han et al., 2021; Yao et al., 2020).

5.2. Influence of loss function on model performance

In the second experiment the influence of the loss function was investigated. Two concepts were explored: a modified version of the balanced loss function as introduced in the original TrajGRU paper by Shi et al. (2017) was compared to a simple RMSE loss, and the application of a mask to the border of model domain was tested. It was found that the balanced loss function increased CSI, POD and FSS-based skillful lead times, but also increased false alarms and MAE and RMSE when compared to the RMSE loss. In the original TrajGRU paper two versions of ConvGRU (Convolutional Gated Recurrent Unit) were compared: one trained with a balanced loss function and one with a regular MAE+MSE loss. They also found the balanced loss function increased CSI. However they did not report on MAE and RMSE.

When exploring the origins of the errors it was found that TrajGRU with RMSE loss makes smaller errors at low rainfall intensities, and TrajGRU Balanced makes smaller errors at high rainfall intensities. A clear trade-off between performance at low rainfall rates and performance at high rainfall rates is visible: (1) If a model makes smaller errors at low rainfall intensities this results in a low total error (indicated by low MAE and RMSE), but also in the failure to detect high rainfall intensities (indicated by low POD, CSI and FSS). (2) If model performance is improved at detecting high rainfall intensities (indicated by high POD, CSI and FSS), this results in a decreased performance at low rainfall rates and increases the total error (indicated by high MAE and RMSE). Franch et al. (2020) also talks about this trade-off in terms of minimizing conditional bias vs. minimizing MSE. In this context, conditional bias refers to the smoothing of forecasts by optimizing on RMSE.

It was found that the application of the mask made a slight difference, resulting in smaller errors. The effect on other metrics was marginal so it could be said that application of the mask is beneficial for model performance. However, the mask significantly reduces the size of the domain of the predictions, which is something that should be taken into account when deciding whether to apply such a mask.

5.3. Comparison to the benchmark

In the third experiment two versions of the TrajGRU models were compared to S-PROG. It was found that TrajGRU trained with the balanced loss function outperformed S-PROG at high rainfall intensities and in terms of CSI and FSS based skillful lead time, but scored worse in terms of RMSE and MAE. Conversely, the TrajGRU model trained with the RMSE loss function outperformed S-PROG at low rainfall intensities and in terms of RMSE and MAE, but scored worse in terms of CSI and FSS based skillful lead time. This means the same trade-off that was mentioned in the previous experiment applies to S-PROG too.

Difference in model behaviour between S-PROG and TrajGRU Balanced was also analysed through two case studies. The event where TrajGRU Balanced significantly outperformed S-PROG, S-PROG predicted dissipation while there was very little dissipation in the observation. The event where S-PROG significantly outperformed TrajGRU Balanced, it correctly predicted the dissipation while TrajGRU Balanced overestimated rainfall intensities.

S-PROG was chosen as the benchmark in this research because of the similarity to the nowcasting system operational in the Netherlands. It is the nature of the model to predict dissipation of the precipitation field at different spatial and temporal scales. This applies especially to high rainfall intensities: high intensity features are smaller, which dissipate faster (Germann et al., 2006). It is thus to be expected that a model like this looses the ability to forecast high rainfall intensities as the lead time increases. There are also other radar extrapolation methods available that are volume preserving and do not predict dissipation. For example in the rainymotion library (Ayzel et al., 2019). These methods have shown to yield longer FSS-based skillful lead times than S-PROG (Imhoff et al., 2020). Since model performance is also evaluated based on FSS-based skillful lead times in this research, it would also be useful to see how the deep learning models presented in this research compare to these models.

5.4. Definition of model performance

So far we have refrained from using the term ’better’ or ’worse’ model performance. This is because the results of the research are ambiguous: what model scores best depends on how model performance is defined. The objective of this research is to accurately predict heavy summer precipitation events. To that end, all models were test on a set of heavy precipitation events. If model performance is then defined by the magnitude of the error, the TrajGRU model with RMSE loss function would be deemed best. If model performance is defined by a models ability to correctly predict high rainfall intensities, scores like CSI, FSS and bias/error at high rainfall intensities become more important and the TrajGRU Balanced would be deemed best. Because of the earlier mentioned trade-off between skill at low rainfall intensity versus skill at high rainfall intensity, there is no model within this research that scored highest on both these levels. How model performance is defined should depend on what the intended application of the nowcasts is. For example, if the nowcasts will be used for hydrological modelling purposes, a metric based on larger spatial scales like the fractions skill score is more important.

5.5. Data pre-processing

The radar reflectivity composites used in this research still contain some clutter. Unvalidated, low level processed data had to be used because of its near real-time availability. A simple pre-processing step of removing all reflectivity values > 50 dBZ removed some of the clutter originating from ship radars and off shore wind farms. However, no attempts were made at removing other types of clutter. Efforts could be made to remove additional clutter, for example by using wradlib, an open-source python library for radar data processing (Heistermann et al., 2013). By removing extra clutter, less irrelevant information is passed to the model. The model does not (or to a lesser extent) have to learn how to deal with the inpredictability of clutter and more learning capacity can remain for the nowcasting task.

Additionally, radar is prone to underestimate rainfall intensity (Holleman, 2007; Overeem et al., 2009). KNMI also provides a QPE which is adjusted with rain gauge data (`https://dataplatform.knmi.nl/dataset/nl-rdr-data-recor-5m-1-0`). An idea could be to include this validation step in the model by using the unvalidated radar reflectivity composites as an input, and using the validated QPEs as the output. This could also improve the applicability of the nowcasts for hydrological purposes.

5.6. Predictability of events

There is a limit to what extent a horizontal representation of the atmosphere can predict what is going to happen next (Germann et al., 2006). This especially applies to heavy summer events. Convective precipitation is a very short lived process: cells dissipate quickly. After rainfall cells have dissipated, the formation of new rainfall cells has to predicted, which is impossible from radar images only and will require a more detailed description of the state of the atmosphere, for example by adding atmospheric variables like temperature, humidity or pressure. In other words, even with the perfect training dataset and the perfect deep learning architecture, model performance will always reach a ceiling because the predictability from radar images only is limited.

6 Conclusions and recommendations

6.1. Conclusions

The objective of this research was to explore the application of deep learning for nowcasting heavy precipitation events in the Netherlands. In this chapter we will see how our research questions can be answered with the results that were obtained during this research.

How can a deep, recurrent, convolutional neural network, be trained to nowcast heavy rainfall events in the Netherlands, using a 13-year radar reflectivity composites archive?

A deep, recurrent, convolutional neural network, based on TrajGRU (Shi et al., 2017) was trained on KNMI radar reflectivity composites. Multiple different training setups were constructed, yielding different versions of the model. All models were tested on a set of heavy summer precipitation events. First, different training datasets were compared. By varying the requirements for valid samples, three training sets were constructed, with different 'dataset size' to 'average intensity of rainfall events' ratios. It was found that a larger dataset results in lower MAE and RMSE. Difference in model performance at other metrics was marginal.

Two concepts in terms of loss functions were tested: (1) the application of a mask to the borders of the domain in the loss function, so the model is not penalized for wrongfully predicting what it cannot yet see, and (2) a balanced loss function where errors are weighted by the rainfall intensity in the observation, to emphasize the importance of the higher rainfall intensities. It was found that the application of the mask resulted in a slightly lower MAE and RMSE. Difference in model performance at other metrics was marginal, so it was concluded that adding the mask is beneficial for model performance. To quantify the effect of the balanced loss function (TrajGRU Balanced), it was compared to a simple RMSE loss (TrajGRU RMSE). TrajGRU RMSE made smaller errors at low rainfall intensities, resulting in the lowest average RMSE and MAE. TrajGRU Balanced however performed better at the higher rainfall intensities (> 5 mm/h). It also outperformed TrajGRU RMSE in terms of CSI at all thresholds and FSS-based maximum skillful lead times. This is a direct result of the definition of the balanced loss functions: errors at small rainfall intensities are tolerated more than errors at high rainfall intensities.

Between the models that were trained, there is no clear winner. There seems to be a trade-off between performance at high rainfall intensities versus performance at low rainfall intensities. What model performs 'best' depends on how the specific task and good model performance are defined. If nowcasting heavy precipitation is defined as correctly nowcasting the high rainfall intensities, then TrajGRU Balanced performs better. If considering larger areas, e.g. city or catchment scale, TrajGRU Balanced also performs better as shown by the FSS scores. But considering that all events in the test set are heavy rainfall events, one could also argue that a lower error over all the heavy events is better model performance, which would mean TrajGRU RMSE is superior.

How does this model compare to S-PROG, a state-of-the-art extrapolation based nowcasting

system?

S-PROG was tested on the same set of heavy precipitation events as above-said models, and results were compared to TrajGRU Balanced and TrajGRU RMSE. S-PROG was outperformed by TrajGRU RMSE in terms of MAE, and performed similar in terms of RMSE. At low rainfall intensities, the error was larger than TrajGRU RMSE, but smaller than TrajGRU Balanced. At high rainfall intensities, S-PROG and TrajGRU RMSE performed similarly, and TrajGRU Balanced outperforms both. In terms of FSS-based scores, S-PROG yielded significantly larger skillful lead times than TrajGRU RMSE, and significantly smaller than TrajGRU Balanced.

The same trade-off that was observed in the comparison between the deep learning models can be observed in S-PROG. It is either better at low rainfall intensities and worse at high rainfall intensities and skillful lead times (TrajGRU Balanced), or the other way around (TrajGRU RMSE). So again there is no 'better' model. However, the deep learning approach is flexible and can yield both a model that is better at low rainfall intensities, or a model that is better at high rainfall intensities. So one could argue that this is the superior approach. The flexibility also gives rise to the idea that in the deep learning approach there is still room for improvement that could be exploited.

6.2. Recommendations

Based on the results of this research some recommendations can be made for further research. It was shown that an increased training dataset size increased model performance. Because in theory, performance a deep learning model keeps increasing with increasing dataset size, it should be explored if more radar data for training could be collected e.g. through transfer learning.

In this research, the deep learning models were only compared to benchmark S-PROG. It is the nature of S-PROG to predict dissipation of the rainfall field, so it did not score well at predicting high rainfall intensities. Many extrapolation-based models do not predict dissipation and it should be investigated if these models can compete with the deep learning approach at high rainfall intensities.

Another recommendation would be to actively take intended end-use into account when defining what model performance is. Forecast verification can be done in many different ways and a clear idea of what is desirable for the end product can help in picking the right metrics, or even when formulating the loss function of a deep learning model.

In this research, little effort was made to remove clutter from the radar data. Data quality is important in data driven processes like deep learning, so it might be useful to provide the deep learning model with higher quality data by more extensive clutter removal, for example with the wradlib library (Heistermann et al., 2013).

Because the predictability of precipitation events from radar images only is limited, especially concerning quickly intensifying and dissipating fields, it should be explored how other atmospheric variables can be incorporated into the model.

A

Appendix A

A.1. Deep learning

A.1.1. Artificial neural networks

One of the most basic forms of a deep learning model is an artificial neural network (ANN). A neuron is a node that transforms one or multiple input value into an output value, using weights, biases and an activation function, see Figure A.1. The activation function transforms the summation of the weighted inputs and the bias, checking whether the magnitude of this summation activates the neuron (output ≠0), or not (output →0). This introduces non-linearity into the model. Without it, the model is just a linear regression model. Common activation functions are sigmoids: projecting the value between 0 and 1, tanh: projecting the value between -1 and 1, ReLU (REctified Linear Unit): if y ≤ 0, output = 0, if y > 0, output = y, and leaky ReLU: similar to ReLU but if y ≤ 0, output = y * -a, where a is a very small value. In Figure A.2 these activation functions are shown.

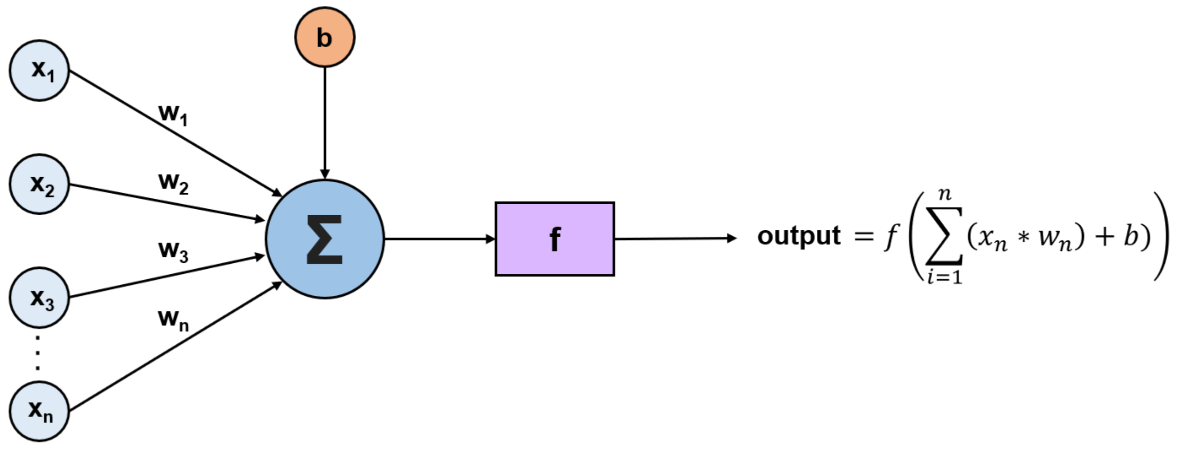


Figure A.1: A single neuron. $x_1 ... x_n$ are the input values, $w_1 ... w_n$ are the corresponding weights, b is the bias and f is the activation function.

An ANN consists of multiple layers of multiple neurons, see Figure A.3. In this Figure the process of training is also described: (1) Feed forward: model output is calculated based on an initial set of weights and biases. (2) Calculate loss: the error in the output is calculated with the loss function. (3) Backpropagation: based on the magnitude of the error, weights and biases of the model are adjusted. This process is repeated until the loss is below a desired value or the model performance stagnates.

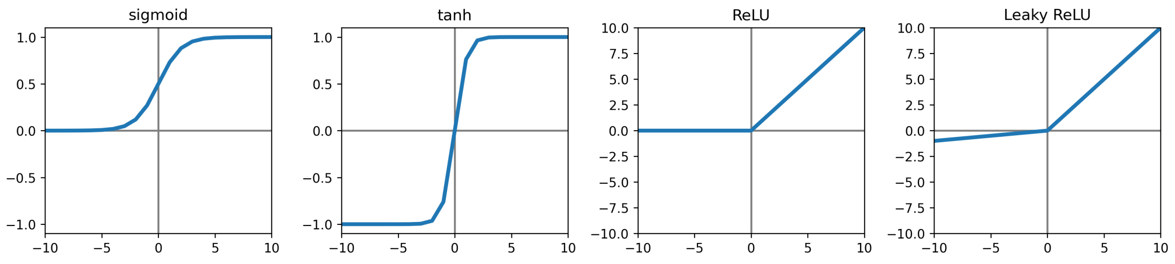


Figure A.2: Four activation functions: sigmoid, tanh, ReLU and Leaky ReLU (with a = 0.1).

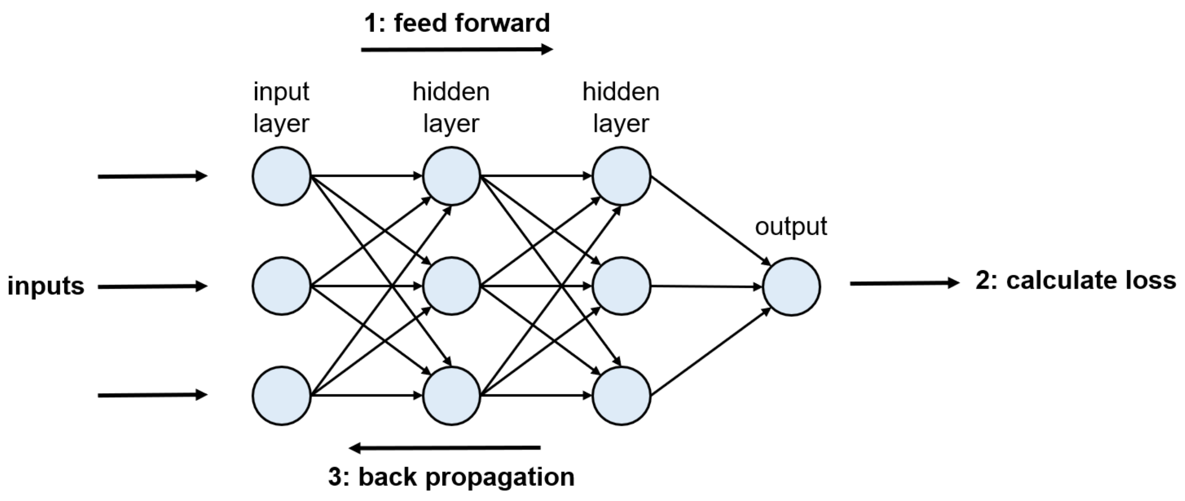


Figure A.3: A multilayer perceptron network.

A.1.2. Convolutional neural networks

A convolutional neural network (CNN) is a type of network that can take images as an input. CNNs make use of kernels (or filters) to extract important features. During training not only weights and biases are adjusted, but also the composition of the kernels. In Figure 3.9 an example of a single convolutional operation is shown. The new image that is created by the convolutional operation is called a feature map. In CNNs, several different kernels are applied to a single input image, creating multiple feature maps (channels).

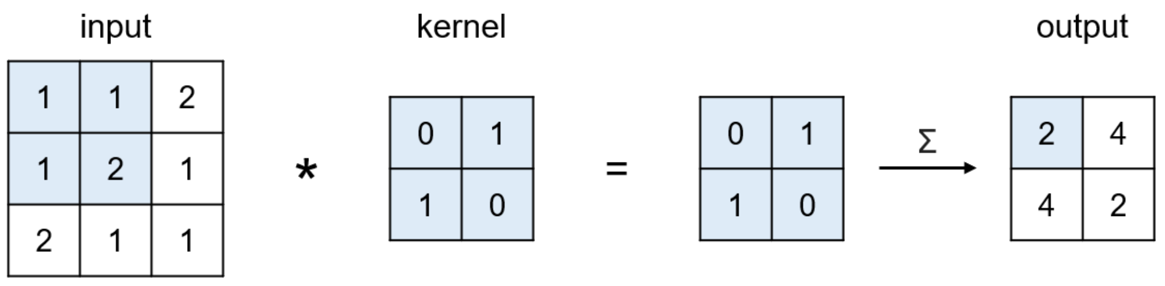


Figure A.4: Example of a convolutional operation with kernel size 2x2 and stride 1.

A.1.3. Recurrent neural networks

A recurrent neural network (RNN) is a type of network that can take sequence data as an input. RNNs make use of recurrent connections, information from previous outputs, also called 'states', for calculating new outputs. These recurrent connections also have weights, which are adjusted during training. Traditional RNNs deal with a vanishing gradient problem: during backpropagation the gradients tend to disappear as it propagates back further in time. This results in the failure to capture long term dependencies. To solve this problem, LSTM (Long Short Term Memory) and GRU (Gated Recurrent Unit) were invented. These are cells that have gates that regulate what information from the past is important and should be kept, and what information should be forgotten. In Figure A.5 the computational flow of a GRU is shown.

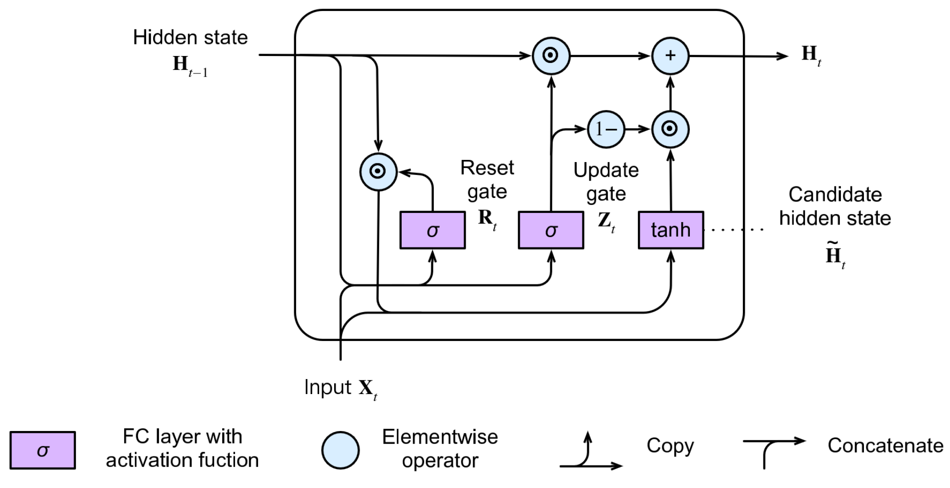

Hidden state $\mathbf{H}_{t-1}$
$\mathbf{H}_t$
1−
+
Reset gate $\mathbf{R}_t$
Update gate $\mathbf{Z}_t$
Candidate hidden state $\tilde{\mathbf{H}}_t$
σ
σ
tanh
Input $\mathbf{X}_t$
FC layer with activation fuction
Elementwise operator
Copy
Concatenate

Figure A.5: The computational flow of a GRU. Adapted from Zhang et al., (2020).

A GRU takes two inputs: the new input X_t and the hidden state from the previous time step H_{t-1}. The update gate Z_t, see Eq. A.1, and reset gate R_t, see Eq. A.2, are vectors with entries between 0 and 1, controlling what information flows through these gates. In Eq. A.3 a candidate hidden state $\tilde{H}_t$ is created by combining X_t and $H_{t-1} \odot R_t$. This means that the reset gate controls to what extend the old hidden state is added to the new candidate hidden state. The new hidden state, see Eq. A.4, is then a combination of the $H_{t-1} \odot Z_t$ and $\tilde{H}_t \odot (1 - Z_t)$. This means that the update gate control how much of the new hidden state is old hidden state, and how much of the new hidden state is the candidate hidden state. This way, the reset gate is responsible for the short term memory of the model, and the update gate is responsible for the long term memory of the model.

$$Z_t = \sigma(W_{xz}X_t + W_{hz}H_{t-1} + b_z) \tag{A.1}$$

$$R_t = \sigma(W_{xr}X_t + W_{hr}H_{t-1} + b_r) \tag{A.2}$$

$$\tilde{H}_t = \tanh(W_{xh}X_t + W_{hh}(R_t \odot H_{t-1}) + b_h) \tag{A.3}$$

$$H_t = (1 - Z_t) \odot \tilde{H}_t + Z_t \odot H_{t-1} \tag{A.4}$$

Bibliography

Ayzel, G., Scheffer, T., & Heistermann, M. (2020). Rainnet v1. 0: A convolutional neural network for radar-based precipitation nowcasting. *Geoscientific Model Development*, *13*(6), 2631–2644.

Bechini, R., & Chandrasekar, V. (2017). An enhanced optical flow technique for radar nowcasting of precipitation and winds. *Journal of Atmospheric and Oceanic Technology*, *34*(12), 2637–2658.

Bowler, N. E., Pierce, C. E., & Seed, A. W. (2006). Steps: A probabilistic precipitation forecasting scheme which merges an extrapolation nowcast with downscaled nwp. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, *132*(620), 2127–2155.

Buishand, T. A., De Martino, G., Spreeuw, J., & Brandsma, T. (2013). Homogeneity of precipitation series in the netherlands and their trends in the past century. *International journal of climatology*, *33*(4), 815–833.

Cao, Y., Li, Q., Shan, H., Huang, Z., Chen, L., Ma, L., & Zhang, J. (2019). Precipitation nowcasting with star-bridge networks. *arXiv preprint arXiv:1907.08069*.

Chung, K., & Yao, I. (2020). Improving radar echo lagrangian extrapolation nowcasting by blending numerical model wind information: Statistical performance of 16 typhoon cases. *Monthly Weather Review*, *148*(3), 1099–1120.

Foresti, L., Reyniers, M., Seed, A., & Delobbe, L. (2016). Development and verification of a real-time stochastic precipitation nowcasting system for urban hydrology in belgium. *Hydrology and Earth System Sciences*, *20*(1), 505–527. https://doi.org/10.5194/hess-20-505-2016

Foresti, L., Sideris, I., Nerini, D., Beusch, L., & Germann, U. (2019). Using a 10-year radar archive for nowcasting precipitation growth and decay: A probabilistic machine learning approach. *Weather and Forecasting*, *34*(5), 1547–1569.

Franch, G., Nerini, D., Pendesini, M., Coviello, L., Jurman, G., & Furlanello, C. (2020). Precipitation nowcasting with orographic enhanced stacked generalization: Improving deep learning predictions on extreme events. *Atmosphere*, *11*(3), 267.

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

Germann, U., Zawadzki, I., & Turner, B. (2006). Predictability of precipitation from continental radar images. part iv: Limits to prediction. *Journal of the Atmospheric Sciences*, *63*(8), 2092–2108.

Han, L., Zhao, Y., Chen, H., & Chandrasekar, V. (2021). Advancing radar nowcasting through deep transfer learning. *IEEE Transactions on Geoscience and Remote Sensing*, 1–9.

Heistermann, M., Jacobi, S., & Pfaff, T. (2013). An open source library for processing weather radar data (wradlib). *Hydrology and Earth System Sciences*, *17*(2), 863–871.

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M., Ali, M., Yang, Y., & Zhou, Y. (2017). Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*.

Holleman, I. (2007). Bias adjustment and long-term verification of radar-based precipitation estimates. *Meteorological Applications: A journal of forecasting, practical applications, training techniques and modelling*, *14*(2), 195–203.

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

Jing, J., Li, Q., & Peng, X. (2019). Mlc-lstm: Exploiting the spatiotemporal correlation between multi-level weather radar echoes for echo sequence extrapolation. *Sensors*, *19*(18), 3988.

KNMI, K. (2014). Climate scenarios for the netherlands. *A guide for professionals in climate adaptation, in, KNMI, De Bilt, The Netherlands*.

Lenderink, G., Mok, H., Lee, T., & Van Oldenborgh, G. (2011). Scaling and trends of hourly precipitation extremes in two different climate zones–hong kong and the netherlands. *Hydrology and Earth System Sciences*, *15*(9), 3033–3041.

Lucas, B. D., & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. *In Proceedings of the 7th international joint conference on Artificial Intelligence*, 121–130.

Marrocu, M., & Massidda, L. (2020). Performance comparison between deep learning and optical flow-based techniques for nowcast precipitation from radar images. *Forecasting*, *2*(2), 194–210.

Marshall, J., Hitschfeld, W., & Gunn, K. (1955). Advances in radar weather. *Advances in geophysics* (pp. 1–56). Elsevier.

Nerini, D., Foresti, L., Leuenberger, D., Robert, S., & Germann, U. (2019). A reduced-space ensemble kalman filter approach for flow-dependent integration of radar extrapolation nowcasts and nwp precipitation ensembles. *Monthly Weather Review*, *147*(3), 987–1006.

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

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

Rauber, R. M., & Nesbitt, S. W. (2018). *Radar meteorology: A first course*. John Wiley & Sons.

Reyniers, M. (2008). *Quantitative precipitation forecasts based on radar observations: Principles, algorithms and operational systems*. Institut Royal Météorologique de Belgique Brussel, Belgium.

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

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

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., & Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, *28*.

Shi, X., Gao, Z., Lausen, L., Wang, H., Yeung, D.-Y., Wong, W.-k., & Woo, W.-c. (2017). Deep learning for precipitation nowcasting: A benchmark and a new model. *arXiv preprint arXiv:1706.03458*.

Simonin, D., Pierce, C., Roberts, N., Ballard, S. P., & Li, Z. (2017). Performance of met office hourly cycling nwp-based nowcasting for precipitation forecasts. *Quarterly Journal of the Royal Meteorological Society*, *143*(708), 2862–2873.

Singh, S., Sarkar, S., & Mitra, P. (2017a). A deep learning based approach with adversarial regularization for doppler weather radar echo prediction. *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 5205–5208.

Singh, S., Sarkar, S., & Mitra, P. (2017b). Leveraging convolutions in recurrent neural networks for doppler weather radar echo prediction. *International Symposium on Neural Networks*, 310–317.

Song, L., Schicker, I., Papazek, P., Kann, A., Bica, B., Wang, Y., & Chen, M. (2019). Machine learning approach to summer precipitation nowcasting over the eastern alps. *Meteorologische Zeitschrift*, *29*(4), 289–305.

Sun, J., Xue, M., Wilson, J. W., Zawadzki, I., Ballard, S. P., Onvlee-Hooimeyer, J., Joe, P., Barker, D. M., Li, P.-W., Golding, B., et al. (2014). Use of nwp for nowcasting convective precipitation: Recent progress and challenges. *Bulletin of the American Meteorological Society*, *95*(3), 409–426.

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

Tran, Q.-K., & Song, S.-k. (2019). Multi-channel weather radar echo extrapolation with convolutional recurrent neural networks. *Remote Sensing*, *11*(19), 2303.

Wang, Y., Zhang, J., Zhu, H., Long, M., Wang, J., & Yu, P. S. (2019). Memory in memory: A predictive neural network for learning higher-order non-stationarity from spatiotemporal dynamics. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9154–9162.

Winterrath, T., & Rosenow, W. (2007). A new module for the tracking of radar-derived precipitation with model-derived winds. *Advances in Geosciences*, *10*, 77–83.

Woo, W.-c., & Wong, W.-k. (2017). Operational application of optical flow techniques to radar-based rainfall nowcasting. *Atmosphere*, *8*(3), 48.

Yan, Q., Ji, F., Miao, K., Wu, Q., Xia, Y., & Li, T. (2020). Convolutional residual-attention: A deep learning approach for precipitation nowcasting. *Advances in Meteorology*, *2020*.

Yao, Z., Wang, Y., Long, M., & Wang, J. (2020). Unsupervised transfer learning for spatiotemporal predictive networks. In H. D. III & A. Singh (Eds.), *Proceedings of the 37th international conference on machine learning* (pp. 10778–10788). PMLR. http://proceedings.mlr.press/v119/yao20a.html

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2021). Dive into deep learning. *arXiv preprint arXiv:2106.11342*.

Zhang, W., Li, W., & Han, L. (2019). A three-dimensional convolutional-recurrent network for convective storm nowcasting. *2019 IEEE International Conference on Big Knowledge (ICBK)*, 333–340.