THESIS

MACHINE LEARNING MODELS APPLIED TO STORM NOWCASTING

Submitted by

Joaquin M. Cuomo

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Summer 2020

Master's Committee:

Advisor: Chuck Anderson
Co-Advisor: V. Chandrasekar

Pallickara, Sangmi Lee
Suryanarayanan, Sid

ABSTRACT

MACHINE LEARNING MODELS APPLIED TO STORM NOWCASTING

Weather nowcasting is heavily dependent on the observation and estimation of radar echoes. There are many different types of deployed nowcasting systems, but none of them based on machine learning, even though it has been an active area of research in the last few years. This work sets the basis for considering machine learning models as real alternatives to current methods by proposing different architectures and comparing them against other nowcasting systems, such as DARTS and STEPS. The methods proposed here are based on residual convolutional encoder-decoder architectures, and they reach the state of the art performance and, in certain scenarios, even outperform them. Different experiments are presented on how the model behaves when using recurrent connections, different loss functions, and different prediction lead times.

ACKNOWLEDGEMENTS

I would like to thank the Fulbright US Student Program for providing the indispensable support to make this happen.

This research is supported by the National Science Foundation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

DEFINITION OF TERMS

ANVIL: Autoregressive Nowcasting Vertically Integrated Liquid method

ar2v: file format for weather data

AUC ROC: Area Under the Curve Receiver Operating Characteristic

AWS: Amazon Web Services

BCE: Binary Cross-Entropy

CASA:Collaborative Adaptive Sensing of the Atmosphere

CNN: Convolutional Neural Networks

CO-TREC: Continuous TREC

convGRU: Convolutional GRU

convLSTM: Convolutional LSTM

convRNN: Convolutiona RNN

CORR: Correlation

CSI: Critical Success Index

DARTS: Dynamic and Adaptive Radar Tracking of Storms

dBZ: decibel relative to Z

DFW: Dallas Forth Worth

ETS: Equitable Threat Score

FAA: Federal Aviation Administration

FAR: False Alarm Rate

GRU: Gated Recurrent Unit

LSTM: Long-Short Term Memory

MAE: Mean Absolute Error

ML: Machine Learning

MS-SSIM: Multi Scale Structural Similarity

MSE: Mean Square Error

NEXRAD: Next-Generation Radar

NOAA: National Oceanic and Atmospheric Administration

NWP: Numerical Weather Prediction

NWS: National Weather Service

POD: Probability Of Detection

PPI: Plan Position Indication

ResConv: Residual Convolutional model

ResGRU: Residual convGRU model

RNN: Recurrent Neural Networks

ROVER: Real-time Optical flow by Variational methods for Echoes of Radar

RR: Rain Rate

S-PROG: Spectral Prognosis

SGD: Stochastic Gradient Descent

SSIM: Structural Similarity

TREC: Tracking Radar Echoes by Correlation

VCP: Volume Coverage Pattern

Z-R: relationship between reflectivity and rain rate

Z: reflectivity

Chapter 1

Introduction

Nowcasting is the forecasting and description of the weather in a short period of time ahead, normally ranging from 0 to 6 hours, and comprising mesoscale features such as individual storms. The main purpose of nowcasting is to make accurate short-term predictions to warn the public, industries, and local authorities about hazardous situations like floods, hail, cyclones, thunderstorms and tornadoes. This way fatalities, accidents, and property damage can be prevented, open-air activities can be planned accordingly, flights and ships can set safe routes, etc. In contrast with forecasting, the short-term window allows meteorologists to harness the highly detailed and continuous information provided by radar echoes not needing to rely on satellite data or time-consuming numerical models. Extrapolating the radar echoes allows having a location-specific forecast of how a storm develops in that area, as it provides information on the shape, intensity, size, direction, and speed of storms.

Nowcasting systems deployed worldwide are currently mostly based on optical flow methods to extrapolate the radar echoes in order to estimate the upcoming weather. However, around 2015 machine learning (ML) techniques started to become more popular in the field of video prediction, which can be directly applied to nowcasting. Some of this work has been done in this specific area but as the author's best of knowledge, no running system based on neural networks is currently being used for nowcasting. The difference of using machine learning compared to the other methods is that it is a physics-free approach, which does not require any prior knowledge about weather physics and it relies entirely on the data fed to the network. Moreover, in the last years, neural networks have proven to outperform many of the traditional approaches in different fields that require expertise in feature extraction, and nowcasting is also showing promising results. Normally, in many fields, machine learning models suffer from the lack of data to be trained which limits their ability to generalize the learning successfully. However, radar data has been recorded for many decades so the problem is not the availability of data but a motivation to harness it.

1.1 Problem statement

The extrapolation of weather radar echoes is the prediction of the distribution, intensity, and appearance of future sequences based on historical observations. This problem can be described as a sequence-of-images to sequences-of-images translation or a spatio-temporal prediction problem.

Given a sequence of n past frames $< f_{t-n}, f_{t-n+1}, ..., f_t >$ it is desired to estimate the k future frames $< f_{t+1}, \ldots, f_{t+k} >$, so the goal is to find a model M such that

$$M(< f_{t-n}, f_{t-n+1}, \ldots, f_t >) \approx < f_{t+1}, \ldots, f_{t+k} >$$

In this work, an event refers to a sequence of images or frames. In Figure 1.1 an extrapolation of input radar echoes is shown and compared to the actual observation.

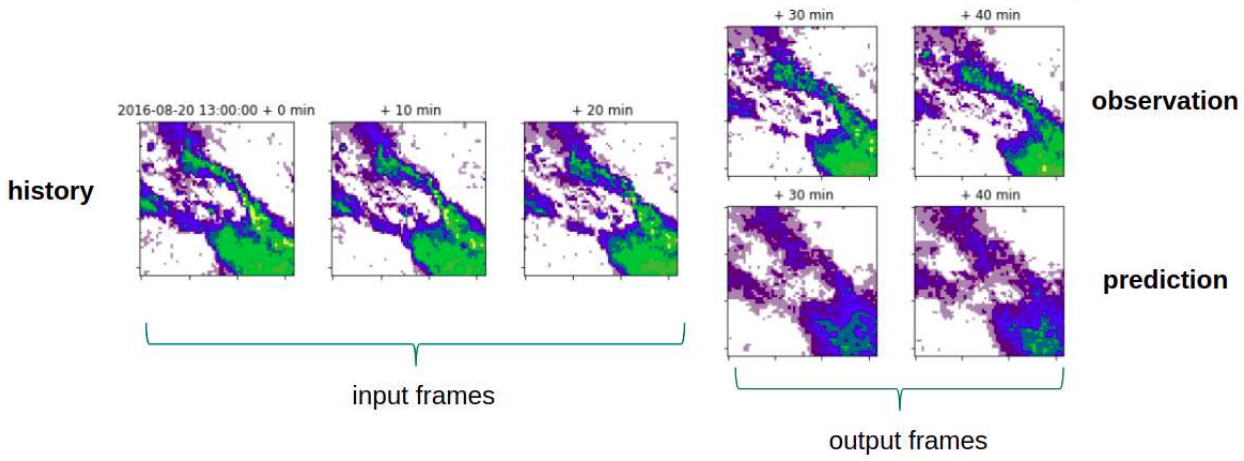


Figure 1.1: Example of a nowcasting systems' input-output.

The parameters involved in this problem are the number of input frames (named history or context), the number of predicted frames, the time increment between frames, and the metrics to evaluate the predictions against the observations (ground truth).

For simplicity in this thesis, the time step between frames is assumed to be constant throughout an event and no time-stamp information is used to make the predictions.

1.2 Research Objectives

The main goal of this thesis is to lay the foundations for future researches using machine learning to create nowcasting models. For this, a thorough literature review is done, and many of the suggestions and models are implemented making a benchmark including not only pre-existing and new ML models but also other nowcasting systems that are currently deployed in meteorological centers.

Secondly, the research is centered on a specific radar network (NEXRAD), and the performance of the models used on it.

1.3 Overview

The thesis is structured as follows. First, an introduction to the problem with all the background information needed to understand this work is presented. Then, the literature review is done where many of the problems to be addressed here are introduced. After that, the methods and experiments along with the results are displayed. Finally, these results and possible research paths are discussed, followed by the conclusions.

Chapter 2

Background information

Precipitation is the result of condensation of atmospheric water vapor falling to Earth's surface in solid or liquid state such as rain, drizzle, snow, graupel, hail, ice pellets, and sleet. It is part of the hydrological cycle, which consists of water going to the atmosphere as vapor due to evaporation or transpiration, the vapor condensates as it gets cooler creating clouds. Once the droplets are heavier than what the vertical motion of the atmosphere can hold, precipitation occurs. There are two basic precipitation generation types which are dynamic (stratiform precipitation) and thermodynamic (convective precipitation). Stratiform precipitation results when a warm front meets a mass of cold air while overriding it and producing extensive horizontal development clouds. The precipitation from this mechanism tends to last longer and to have less intense rain as well as a uniform intensity. Convective precipitation occurs when localized air is heated more than the surroundings creating instability. As the density of the air mass diminishes, it rises and gets cooler, forming clouds with vertical development called cumuliform. Normally, the type of rain it produces is short and localized with rapid changes of intensity.

Meteorology is a branch of atmospheric sciences that studies weather processes with a focus on forecasting. To do so, they capture and analyze different variables to create a model that can represent the behavior of the atmosphere, oceans, and land surface so future events can be predicted. Those variables can be obtained by a suite of observing systems distributed worldwide such as satellites, radars, and surface weather observations. Commonly, for time scales of several hours or days, a numerical weather prediction (NWP) method is used, which consists of creating a three-dimensional grid of the atmosphere and computing physical equations that can describe it. On the other hand, short term predictions of a few hours are normally done by extrapolating current weather trends using only statistical methods rather than physics-based ones.

Precipitation estimation is done by measuring rain rate (RR), which represents how much rain is falling in height by time, normally expressed in mm/hr. There are different ways to estimate

RR, where measuring directly at the surface with a rain gauge is the most accurate. Nevertheless, it is possible to use radar echoes to estimate the current rainfall. The radar measures the presence of hydrometeors, which are any product of water vapor condensation or deposition, by means of the reflectivity factor Z (explained in more detail in the upcoming paragraphs), which can be converted to rain rate (RR) by using an empirical relation Z-R. This method has certain limitations due to calibration errors, overshooting of the radar beam (when the beam becomes more elevated above the Earth's surface while increasing range), and the complexity of the Z-R relation (different drop size distribution can yield same reflectivity but different rain rate) [1], but it has considerable practical use to remote estimate rain rate and more importantly to make extrapolations in the future.

A weather radar progressively emits a constant wavelength beam at different elevation angles and every direction by rotating 360 degrees. However, it does not emit parallel to the ground nor perpendicular. Each of these sweeps is called a Plan Position Indication (PPI) scan and the accumulation of all PPIs is the Volume Coverage Pattern (VCP). The energy emitted can be dissipated, absorbed, or reflected. When some of the emitted energy hits a target a portion is reflected back to the emitter called the echo and is sampled by the radar to obtain information about how far and how big the target is. The radar echoes correspond to obstacles from a certain volume as the beam expands with distance from the radar. This volume, called gate, depends on the specifications of the radar, such as beam wavelengths and pulse repetition time, and it can be thought of as the pixel size of the resulting data matrix. As the volume increases with the distance from the radar, the rainfall intensity is increasingly underestimated, the rainfall coverage is increasingly overestimated, and obstacles present in only a portion of the volume are averaged over the whole volume. Thus, the resolution is not constant and decreases with distance.

The reflected power can be interpreted as what is called base data or composite data. The former consists of a single scan of the radar while the latter is the highest measurement within a vertical column. Because water can evaporate at higher elevations, only the lowest scan is used for estimating the rain rate, while for other purposes another scan and composite data are more suitable to analyze.

The most important measurement from the radars is the reflectivity factor or Z factor. Normally, it is expressed in decibels (dBZ). As the range of Z goes from 0.001 mm^6/m^3 (caused by light fog) to 36,000,000 mm^6/m^3 (large hail), dBZ goes from -30 dBZ to 75 dBZ [2]. Z is computed as the sum of the sixth power of the hydrometeor diameter overall hydrometeors in a unit of volume:

$$Z = \int_0^\infty N(D)\ D^6\ dD \tag{2.1}$$

where D is the diameter of the hydrometer in mm and N is the distribution function of the hydrometers.

To get to the Z-R relationship mentioned above the most common drop size distribution used is the Marshall-Palmer which fits the data to an exponential function, with N_0 and Λ as constants.

$$N(D) = N_0 e^{-\Lambda D} \tag{2.2}$$

and the rain rate (R) is measured as depth of water per unit of time,

$$R = \frac{\pi}{6} \int_0^\infty D^3\ N(D)\ w_t(D)\ dD \tag{2.3}$$

where w_t is the terminal velocity.

From both equations 2.1 and 2.3, the following relationship can be obtained:

$$Z = aR^b \tag{2.4}$$

Unfortunately, the parameters a and b cannot represent accurately every type of precipitation. Numerous studies have tried to establish common parameters between different types of rains (stratiform, thunderstorm, ice, snow), but no prevalent parameters fit best as they widely vary within different rains and different places [3]. Nevertheless, having the reflectivity information is extremely useful to predict rain.

Chapter 3

State of the Art

Nowcasting methods have been evolving since the earliest subjective interpretation of what could happen in the next few hours to modern methods that try to automate the predictions using models of different complexities. They could vary from simple extrapolations of radar echoes to more elaborate and generally more accurate systems combining multiple observation systems such as satellites, radars, and surface stations. When the time-lapse is short and the scale is small, extrapolation techniques are the most frequent choice. As shown in Table 3.1, radars are the main observation type that can be used to issue warnings. For this reason, the following review of nowcasting techniques is focused on using radar data.

Nowcasting systems needs to run in real-time, thus complex models often do not perform well, especially NWP where their normal update time is 3 hours or more, and the meteorological parameters can have a high sampling frequency such as 1 minute for radars. However, there are adaptations of NWP that are intended to perform faster. Due to the complexity of finding global parameters for weather, many of the methods used have different advantages and limitations and they normally perform better for the region and the main goal they were developed for, such as issuing warnings for flood or tornados. Table 3.2, taken from the 2017 version (last available) Guidelines for Nowcasting Techniques by the World Meteorological Organization [4], lists some of the operational systems divided into different categories according to how they approach the predictions. Some methods first detect the precipitation area and then track it, while others extrapolate the whole radar image. It is common for the latter ones to use optical flow methods to obtain the motion vectors and then apply semi-lagrangian advection schemes to make the prediction. More complex methods integrate several inputs and different nowcasting algorithms. It is worth remarking that no machine learning approach is listed by the time the Guidelines for Nowcasting Techniques was written.

Table 3.1: Warning observation types. Table taken from [4].

Weather phenomena	Surface stations	Dense Surface Station	Upper-air observations	Satellite	Lightning detection	Radar	NWP global/ regional
Thunderstorm location	4	2	5	1*	1	1	5/3
Hail size	5	2	3	3	5	1	5/4
Tornado	6	5	5	4	5	1	6/5
Microburst	5	2	5	4	5	1	54/4
String surface wind	4	2	3	4	5	1	5/3
Heavy precipitation: flash flood rain (large scale) snow (large scale)	 4 4 4	 1 1 2	 5 5 5	 3 2 2	 5 5 5	 1 1 1	 5/3 3/2 3/2
Precipitacion type (winter)	2	1	2	3	6	1	5/2
Visibility/fog	2	1	5	4	6	2**	5/3
Surface icing by freezing precipitation	2	1	2	4	6	2	5/3

Notes:
1 Can often be used to issue warnings.
2 On limited occasions can be used to issue a warning.
3 Can often be used to issue a watch.
4 On limited occasions can be used to issue a watch.
5 Only useful when combined with other observations.
6 Limited usefulness even when combined with other observations.
* While warnings can be made on thunderstorm location, the location is generally not as specific as with radar or lightning.
** Only when restricted by heavy precipitation.

After applying one or more of these algorithms a human nowcaster should examine and process the results with other information, such as likely changes that could occur during the day to decide if severe weather may happen.

The evaluation of the nowcasting algorithms is not a straightforward task. Assessing the quality of a prediction depends on the original intention of it and the nature of the variables being used. As mentioned before, human evaluation of the nowcast is critical to delivering a final report, thus visual inspection is essential when thinking on the final user. Nevertheless, a quantitative assessment should be performed to provide scientific verification. Unfortunately, no single verification metric can assess all the attributes to provide an overall performance score. For example, for continuous

Table 3.2: Literature summary on prediction models from [4] categorized by their approach.

Type of system	Acronym (name)	Main reference
Cell tracker	TITAN (Thunderstorm Identification Tracking Analysis and Nowcasting)	Dixon and Wiener (1993) [5]
	SCIT (storm cell identification and tracking)	Johnson et al. (1998) [6]
	TRT (thunderstorm radar tracking) algorithm	Hering et al. (2004) [7]
	FAST (fuzzy logic algorithm for storm tracking)	Jung and Lee (2015) [8]
Area tracker	CO-TREC (continuous tracking radar echoes by correlation)	Li et al. (1995) [9]
	MAPLE (McGill algorithm for precipitation nowcasting by lagrangian extrapolation)	Germann and Zawadzki (2002) [10]
	Optical flow in the GANDOLF (Generating Advanced Nowcasts for Deployment in Operational Land-based Flood Forecasts) system	Bowler et al. (2004) [11]
	DARTS (Dynamic and Adaptive Radar Tracking of Storms)	Ruzanski et al. (2011) [12]
Multiple observation system	ANC (Auto-Nowcast) system	Mueller et al. (2003) [13]
	SWIRLS (Short-range Warnings of Intense Rainstorms in Localized Systems)	Li and Lai (2004) [14]
	NowCastMix – Autowarn (nowcast mix – automatic warning)	James et al. (2015); Reichert (2009) [15]
	CAN-Now (Canadian Airport Nowcasting) system	Bailey et al. (2009) [16]
	INCA (Integrated Nowcasting through Comprehensive Analysis)	Haiden et al. (2011) [17]
Probabilistic approach	STEPS (Short-term Ensemble Prediction System)	Bowler et al. (2006) [18]

values of the forecast mean absolute error and mean squared error are good accuracy measures. On the other hand, if the prediction is binary the variations of precision and recall could be used. Finally, if there are multiple categories or the nature of the forecast is probabilistic, contingency tables and AUC ROC (Area Under the Curve Receiver Operating Characteristic) are common metric approaches.

In the last 4 years, with the increased availability of data and computing power, machine learning approaches started to become more popular in the area of weather nowcasting. Some papers have already shown machine learning approaches that outperform optical flow based methods such as ROVER [19], TREC and CO-TREC [20], and Farneback [21]. More recently, new techniques [22] are being developed with promising results to overcome the main drawback of these methods which is the blurry effect on the predictions [19, 23–28].

Nowcasting can be framed in a more general formulation as a video prediction problem, which has been a matter of study in the computer vision field for many years. They are both showing similar paths, as they started using optical flow methods and now, they are moving to machine learning approaches. Particularly, an important breakthrough for both communities to start looking at ML

was the model proposed by Shi et al. in 2015 [19] where they combined the Recurrent Neural Networks (RNN) largely used to model sequences with Convolutional Neural Networks (CNN), well known to successfully capture spatial information in images. Since then, many other works have been published in the area. Like non-ML approaches, each of them addresses a particular problem and no consistency on datasets for benchmarking has been used. The most common dataset is the moving-MNIST, which consists of 2 or 3 digits (moving-MNIST++ version) bouncing in a square two-dimensional box. However, this dataset does not capture all the nuances a radar echo sequence can have, such as rotations, changes in intensity, and shape, among many others. Also, most of the papers use different radar technologies, from different locations, with different sampling rates, and they also combine with other data such as wind velocity. With respect to the metrics used for evaluating the performance of the models, the most common is a combination of Mean Square Error (MSE), critical success index (CSI), false alarm rate (FAR), probability of detection (POD), and correlation coefficient (explained in detail in Section 4.3). Finally, there is a disparity in how many frames are used as input and how many are predicted. It varies from 4 to 10 for the input and 1 to 20 for the prediction.

In Table 3.3 a list of different publications using machine learning approaches to address weather nowcasting is shown. In blue are marked those which are more relevant to this thesis mainly for the type of input data as well as the metrics used.

The first two published papers using machine learning for weather nowcasting were from the creators of the convLSTM layer [19,29], in 2015 and 2017 respectively. The architecture used consists of a symmetrical encoder-decoder using convolutional recurrent layers, predicting 10 frames. It was trained on reflectivity values from the Hong Kong area. In the second publication they even proposed this dataset to be a standard benchmark for ML in nowcasting, but with low success as it is not extensively used by the community. The results from their models were the kick start to all the upcoming papers in this topic as they outperformed the optical flow based model ROVER [30], created by the same group, and two ML approaches, one using non-convolutional recurrent layers [31] and the other one using only convolutional layers [32].

In 2018, [20] proposed Recurrent Dynamic CNNs (RDCNN). It combines convolutional layers with recurrent structures and a probability prediction as described by the authors. It was compared against the CO-TREC model and it achieved better scores at every lead-time. The data used was from three different areas in China, and the model uses four history frames and it predicts only one. In the same year, a modification of the PredNet [33] model created for general video prediction was applied by [34] to weather nowcasting. They used data from Kyoto, and the model was evaluated against the trajGRU model proposed by [29], where the results were in favor of PredNet in most of the experiments, except at high reflectivity values. PredNet uses a different type of encoder-decoder architecture, where the error between the prediction and the target is not only backpropagated by the optimizer but explicitly used as the input to recurrent layers. The model was adapted to accept ten input frames and predict the future ten frames.

The model AENN proposed by [35] in 2019 has the novelty of using a Generative Adversarial Network for generating the predictions of radar echoes. They compared the model against convLSTM [19], ROVER, and TREC using data from five Chinese regions. The results showed that AENN did better in most of the metrics, as well as it produced more realistic predictions when visually inspected. In this case, the model takes 5 input frames and predicts the next 3. Other paper came out that year ([24]) doing a comprehensive analysis on different loss functions and introducing Image Quality Assessment metrics to weather nowcasting using [19, 29] models. With a different dataset from the one used in the original papers, they obtained better results using convGRU rather than trajGRU as [29] showed, but after some modification on the convolutional filter sizes they did achieve the best results with the named dec-trajGRU. Another difference in the comparison was the length of the input data, which was 5 instead of 10, though the prediction remains in 10 frames.

In 2020, the last proposed model using ML for weather nowcasting while this thesis is being written is RainNet. The assessment was not done with another ML approach but with the same author's RainyMotion [36] optical flow based-model, for which the results were in favor of RainNet.

Table 3.3: Literature review of machine learning models used for weather nowcasting.

Proposed	Reference	Compare to...	Datasets	Training loss	Evaluation metrics	Results	In/Out
convLSTM	[19]	FC-LSTM ROVER	moving-MNIST Hong Kong Z (rainy days)-prepocessed	MSE-MAE	MSE / CORR CSI / FAR / POD	outperform in all	10/10
convGRU trajGRU	[29]	conv3D conv2D DFN ROVER	moving-MNIST++ HKO-7 - preprocessed	MSE-MAE B-MSE/MAE	B-MAE / B-MSE HSS / CSI	outperform in all	10/10
RDCNN	[20]	COTREC	3 from CINRAD-SA	NM	MSE / CORR CSI / FAR / POD	outperform in all	4/1
SDPredNet	[34]	trajGRU	moving-MNIST++ KyotoYahoo! Static Map API	B-MSE	MSE HSS / CSI	outperform in many	10/10
AENN	[35]	convLSTM TREC Optical Flow	CINRAD/SA	cross entropy	CSI / POD FAR / HSS	outperform in many	5/3
dec-trajGRU convGRU convLSTM	[24]	trajGRU convGRU convLSTM	moving-MNIST++ CIKM AnalytiCup 2017 (3.5km)	MSE / MAE MS-SSIM / SSIM	MSE / MAE / CORR SSIM / MS-SSIM CSI / FAR / POD	outperform in all	5/10
U-Net	[37]	MRMS Optical Flow Persistance	MRMS dataset	cross entropy	Precision-Recall	outperform in short time	NM
RainNet	[38]	Rainymotion Persistance	Germany DWD data	LogCosH	MAE-CSI	outperform in all	4/1
LSTM- PERSIANN	[21]	Farneback Persistency RAP RNN	IR of GOES Climatic Prediction Center	NM	MSE / CORR CSI / FAR / POD	outperform in many	NM/12 (6hours)
SBOW (SVM)	[39]	TITAN	NEXRAD+VDRAS	NM	CSI / FAR / POD	mixed	NM
DozhdyaNet	[40]	Optical Flow Persistence	RY product of the German Weather Service	NM	MAE / CSI	worst	NM
MLP	[41]	Persistance	Swiss C-band radars	NM	Predicted Quantile / RMSE	better	nm
convLSTM	[42]	TREC Optical Flow	RGB echo + (U,V) wind	NM	MSE / CORR CSI / FAR / POD	outperform in all	5/20

The data used is not raw reflectivity values, but a quality-controlled rainfall-depth composite of 17 radars. The model takes in 4 frames and outputs 1.

Outside weather nowcasting, in the last two years more than twenty papers have been published using machine learning for video prediction [43–54] showing the relevance of the task for the computer vision community. Many of these papers are modifications of other models while others are introducing new approaches [55–61].

It is important to remark on some of the differences with many video prediction framings when trying to transfer the techniques to weather nowcasting. First, the sampling rate is not always constant when dealing with radar echoes. Second, the objects that are being tracked are the accumulation of hydrometeors, whose shapes continuously change. Moreover, these shapes can grow and shrink, disappearing, and reappearing in the radar echoes, especially if only one elevation scan is used. Even the change of intensity is important to consider as a variable function as the assump-

tion to be constant is commonly used in optical flow approaches as they try to track one pixel from a frame to the other based on the initial brightness.

As this work was inspired by the Convolutional Recurrent Neural Networks (convRNN), which was developed by the nowcasting community, more details on this network are given below.

The basic concept behind a Recurrent Network is that they keep as a 'hidden' state a weighted sum of previous outputs, being able to model sequences. One major improvement in RNN was the Long-Short Term Memory RNN (LSTM) [62], which introduced different gates to have more control over how much the network should remember and how much it should forget. Then, [63] proposed a simpler model with fewer gates called Gated Recurrent Unit, which shares the same concept but uses only one gate for both remembering and forgetting. Finally, the adaptation of the RNN to a convolution RNN is done by changing the fully connected operations by convolutional operations [19]. Some variations of convRNN have been proposed such as trajGRU [29] and rgcLSTM [59] which have shown good results.

Chapter 4

Research Methodology and Experiments

This Chapter is structured as follows. The first part is about the data used to train and test the models. Then, the proposed models are described. After that, the metrics to measure their performance and the baseline models are introduced. Finally, the experiments to assess the proposed models are explain.

4.1 Dataset

The datasets used to train, evaluate, and test the different models use NEXRAD data, because of its convenience (AWS storage) and availability (several years and regions). In the U.S. territory, the NEXRAD (Next-Generation Radar) net of 160 radars deployed throughout the country provides cost-free public access to radar data. The system is controlled by the National Weather Service (NWS), the Air Force Weather Agency, and the Federal Aviation Administration (FAA) and it is administered by the National Oceanic and Atmospheric Administration (NOAA). The data consists of base data and derived products, which are called Level-II and Level-III, respectively. Level-II includes reflectivity, mean radial velocity, spectrum width, dual-polarization base data of differential reflectivity, correlation coefficient, and differential phase.

For machine learning the data is critical, as it is the only resource for it to learn. Therefore, below is an explanation on how the data was selected and processed from the raw files downloaded from AWS to the final datasets files.

Pre-processing of the dataset

To transform individual ar2v files (one per radar scan, meaning one for each timestamp) into sequences of N frames the following steps were followed.

First, the reflectivity data was extracted from the ar2v files. Then, converted from polar coordinates to Cartesian coordinates using PyArt [64] gridding function with grid limits of 300.000

km by 300.000 km by 20 km (latitude, longitude, altitude). Also, the gridded map was defined in a 64 by 64 matrix. Afterwards, the reflectivity values were clipped from 0 dBZ to 70dBZ and then scaled from 0 to 255 (pixel values). Before saving the matrices as PNG files the missing values were filled with 0 dBZ. The sample rate for each scan is not perfectly constant, and to avoid discarding sequences with missing scans, a tolerance of 20 minutes was used to consider it consecutive. Finally, the frames were re-arranged in same length sequences thus creating a list of events. The choice of the frames being on gray scale and with a 64x64 resolution was taken to reduce memory requirements during the training of the models.

Post-processing of the dataset

As most of the dataset contains frames with low reflectivity it was reduced to have only relevant events. To consider an event as relevant a kernel was convolved on each frame to detect areas of high intensity and the event was classified as relevant if it contained more than half of its frames with at least one pixel exceeding a threshold. The threshold was set to 30 dBZ and the size of the kernel was an eighth of the frame size. Figure 4.1 illustrates how the processing was done.

After filtering out the non-relevant events the datasets were reduced approximately to a fifth of its size. For the training and validation set that was all the post-processing done. For testing, two different datasets were used, and a hand-picked selection of the most relevant events was done (prior to any evaluation on them) based on the presence of different advection behaviors, discarding out those with reduced areas of high reflectivity and receptivity. The reason for this further reduction was to minimize bias by having many events with similar characteristics and to reduce both memory consumption and external hand-labor to be able to evaluate all the events with all external nowcasting methods. From the raw data to the processed training dataset the size reduction was from 4 TB to 200 MB.

The sampling rate of the frames is a limitation that has not been addressed in depth. The training set has mixed events with frames spaced from 4 to 6 minutes apart. Within each event, the spacing remains constant, but because the ML models do not take into account the time-stamp the models are not being trained to predict a fixed time into the future. The reason behind why

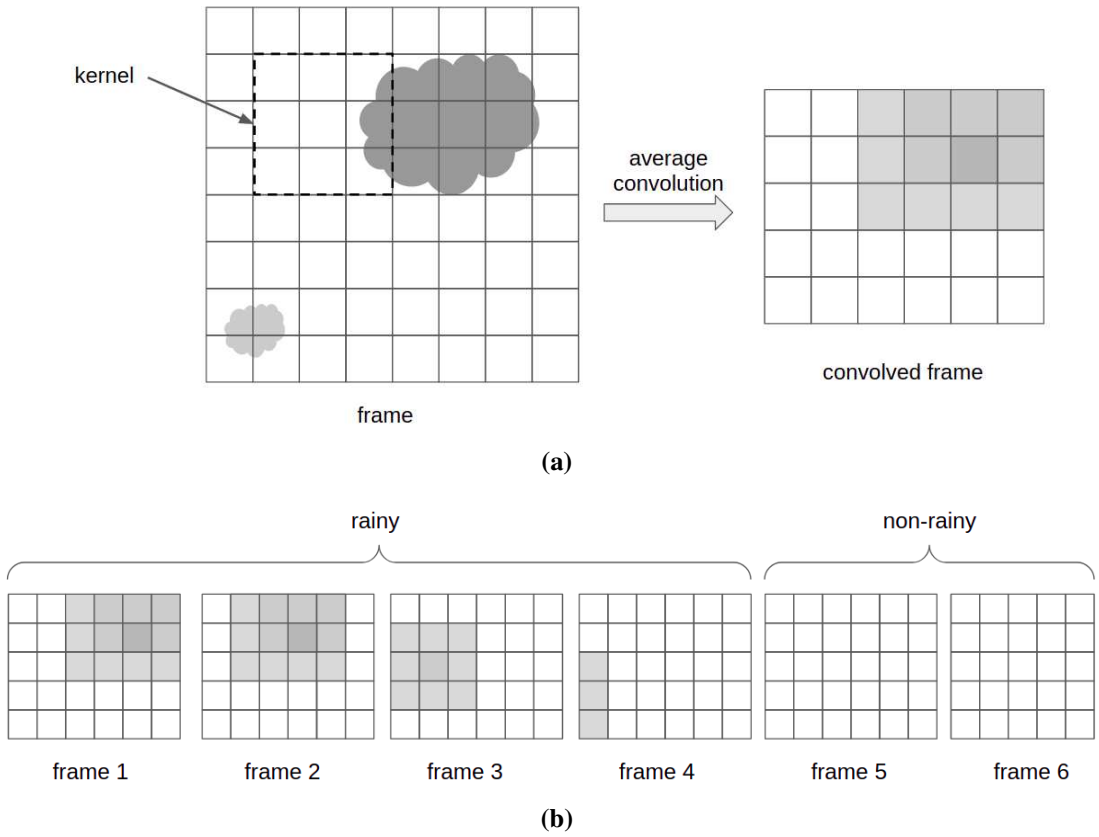


Figure 4.1: Example on how a sequence is processed to consider it or not a relevant event. (a) shows the kernel applied to each frame and its output. (b) shows the entire sequence and in this case, as there are more 'rainy' frames than 'non-rainy' the sequence is relevant and included in the training dataset.

this has not been analyzed/treated with more depth was due to the time-spaced-requirements of having larger datasets and the availability of them. Hopefully, a potential advantage of having this heterogeneous sampling rate is that the models will generalize better the dynamics of the observed frames and make the predictions within the same time-scope. Nevertheless, some of the baseline models used in the benchmark rely on a strict step size. Therefore, one of the two testing datasets was re-sampled to have an even time lapse between each frame as required by some of the models. The re-sampling was done in Cartesian coordinates using linear interpolation from a ~4 minutes spacing to 5 minutes. It is important to remark that most likely doing the interpolation in the polar coordinates on each scan ray would be more accurate or to apply an advection correction in the Cartesian coordinates.

In Figure 4.2, a scheme of how each dataset was constructed as well as the name for which they will referenced in this thesis. First, it shows the location and time, and then what type of post-processing was applied.

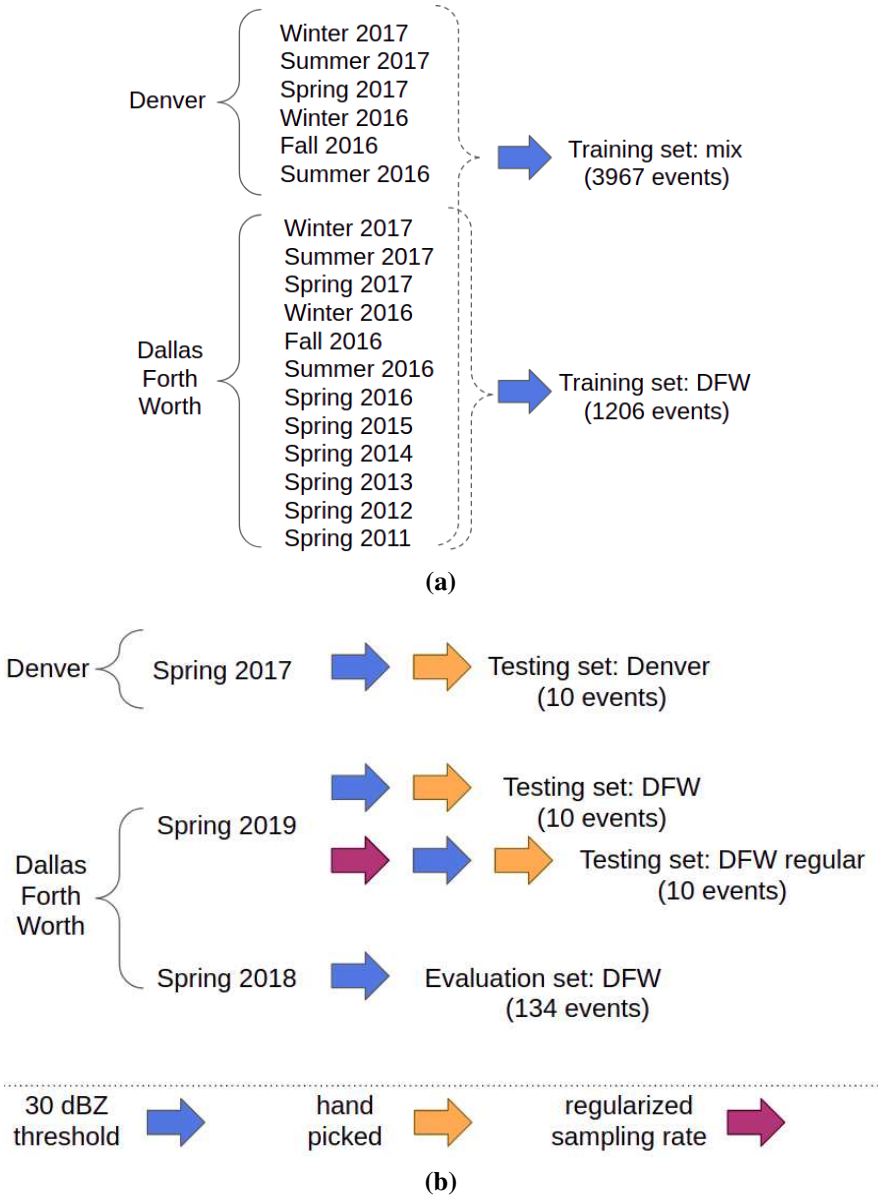


Figure 4.2: Diagram on how the datasets were created, (a) shows the training datasets and (b) the validation and testing datasets. The 30 dBZ threshold arrow refers to the process explained in Figure 4.1, the yellow arrow means that visual inspection and a manual selection was done, and the red arrow refers to the procedure described in the above paragraph.

4.2 Models

Throughout this thesis different machine learning models are used and compared with each other with the goal of laying the foundations for further research on this line of work. Therefore, not only the models with the best performance are shown but also those using techniques mentioned in the literature or different original ideas. All of them share some core characteristics. Firstly, the training is done in an unsupervised fashion, meaning that it does not require labeled data. Figure 4.3 shows how the dataset containing only sequences of reflectivity data is used for both the input and the target values of the model. As the goal is to predict future frames, the first part of the sequence is the context used to predict future frames, and the second half is used to calculate the accuracy of the prediction. The unsupervised training is to some extent inherited from the nature of the problem formulation in this thesis, but it could be other cases such as when using ML estimations of optical flow to extrapolate future frames, in which labeled data is required. Secondly, all models have the same structural architecture of an encoder-decoder. Conceptually, this refers to when the input data is encoded into a state, in this case of lower dimensionality, and then the output is reconstructed using, sometimes, only that state.

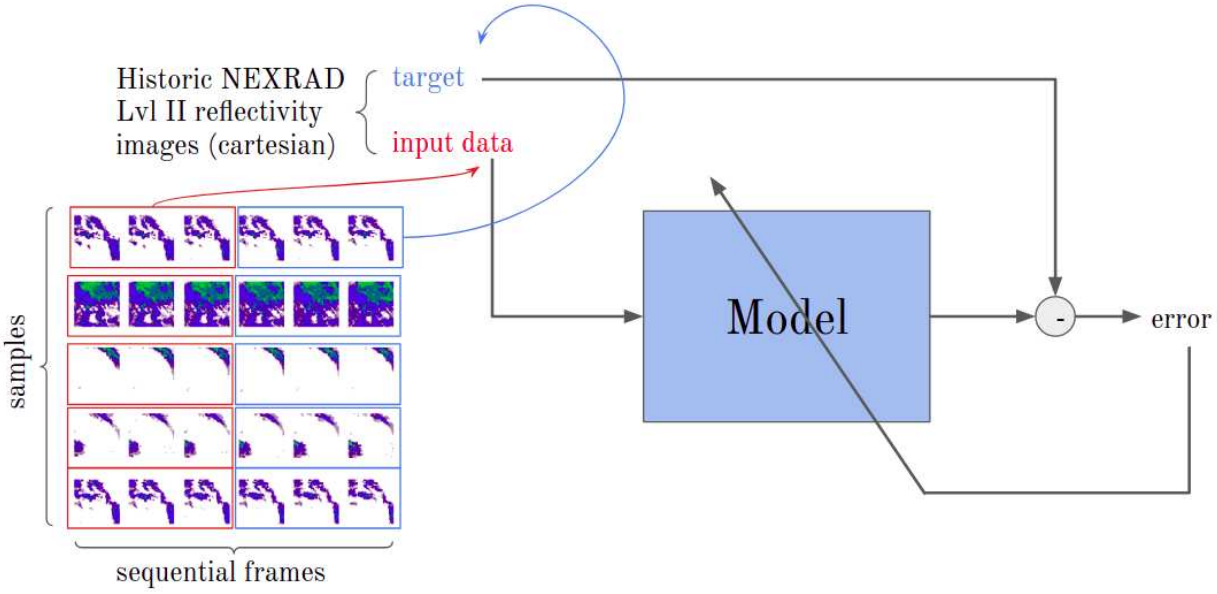


Figure 4.3: Diagram on how an unlabeled dataset is used in unsupervised training

The model variations addressed in this thesis can be grouped in different architectures, loss functions, and types of outputs. Figure 4.4 shows each of these groups which will be developed in detail the following paragraphs. Conceptually, the architecture group refers to the type of layers being used, the type of output (or prediction) to what is being predicted (what target is being used), and the loss functions to which performance metric the optimizer uses during the training.

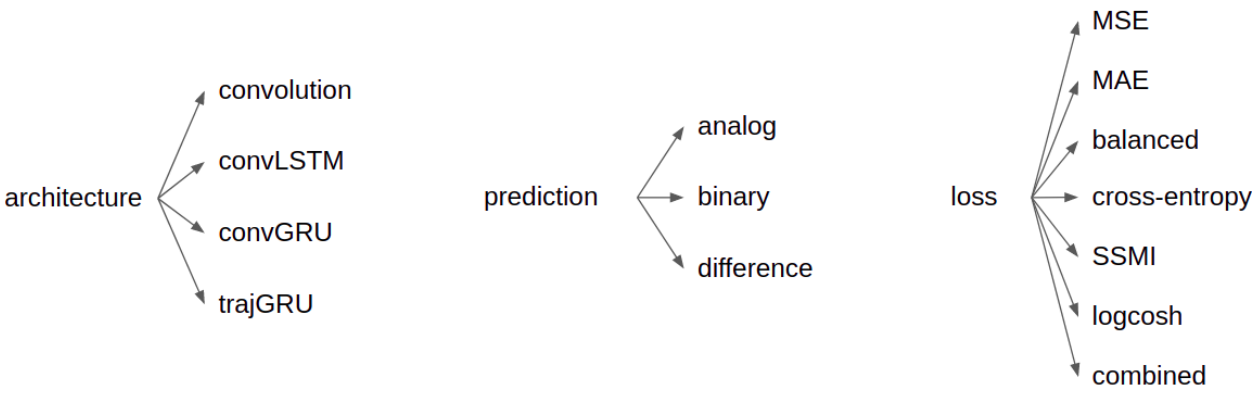


Figure 4.4: Variations analyzed in the models.

As mentioned before the architectures always follow an encoder-decoder structure, but they differ in the number and type of layers used. Two main categories can be distinguished, purely convolutional layers, or a combination of convolutional with recurrent layers (LSTM, GRU, traj-GRU). Also, for every model the option of using skip-connections has been tested [65]. For details on each implementation refer to the Appendix A.6 and the code [66]. The two most relevant models are diagrammed in Figures 4.5 and 4.6.

Model ResConv

Model ResConv is shown in Figure 4.5. The convolution layers are of kernel size 3, strides of size 2, and padding of size 1 for each dimension. The downsampling is done using a convolution of kernel 4 and strides 2. While the upsampling has the same parameters, it is a transposed convolution (also known as fractionally-strided convolution or deconvolution). For the convolution layers in the decoder section, the transposed convolution is also used.

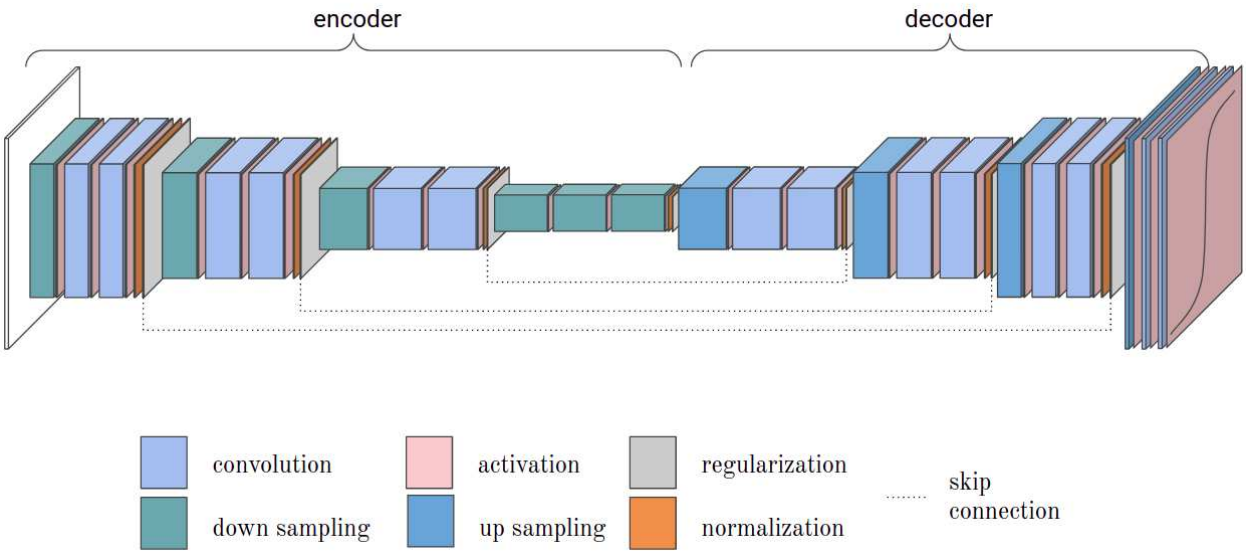


Figure 4.5: Diagram of residual convolutional model.

The activation layers are Leaky-Relu except for the last one which is a Sigmoid. The regularization is done using a Dropout layer with a rate optimized for each case, and the normalization is a Batch Normalization layer. Finally, there are residual connections (skip connections) between the encoder and the decoder, where the output of the decoder is added to the output of the decoder layer with same dimensions.

Model ResGRU

This model, shown in Figure 4.6, unlike the one proposed at [29] not only differs slightly in the structure, but it has residual connections between the encoder and the decoder. Also, the convolutional layers are a compound of many layers following the logic of the residual networks. Following the design in [29], the initial states of the recurrent networks in the decoder layers, in this case convolutional GRUs, are given by the output state of the encoder layers.

The names given to these models are just for ease of reference throughout this thesis.

The second group distinguishes the models on what type of output they predict. The most forthcoming approach is to predict the desired output frames, meaning that the pixel values vary between 0 dBZ to 70 dBZ continuously. Because many of the standard metrics for weather now-

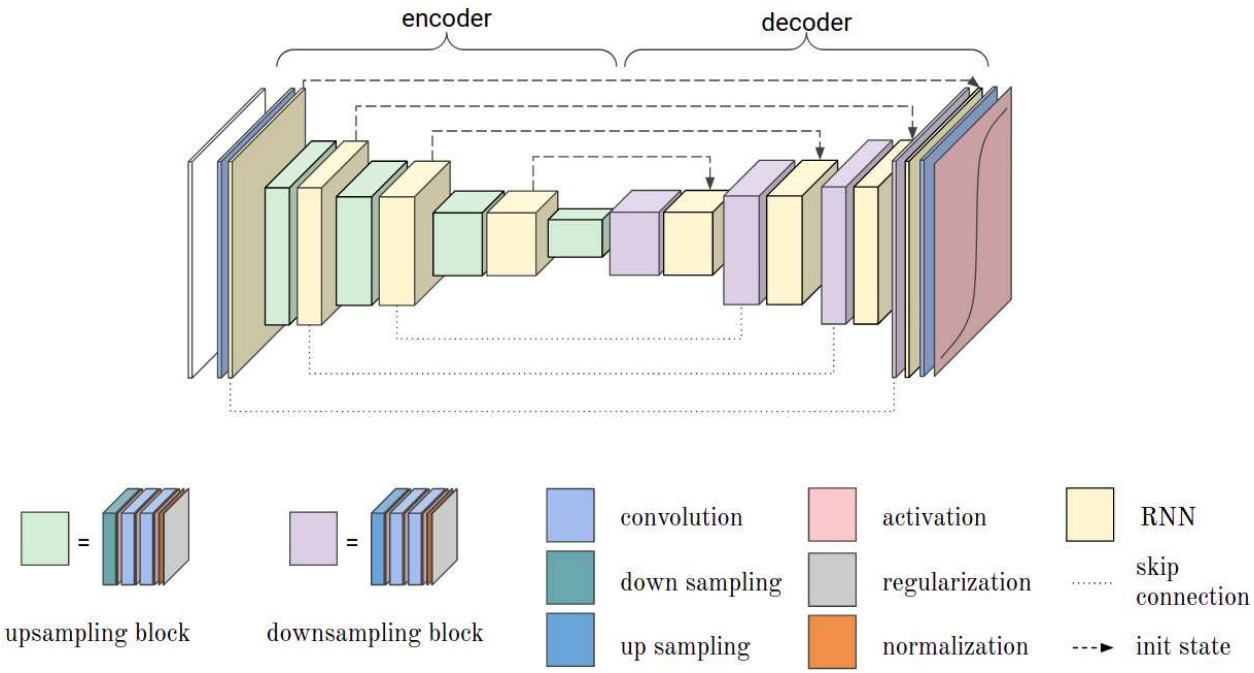


Figure 4.6: Diagram of residual convolutional recurrent model.

casting require the reflectivity values to be binarized using a threshold, it was also attempted to directly predict the binary values. In this case, the input remained continuous with the aim of anticipating growths and decays on the values. Finally, it was also tried predicting only the difference between the previous frame and the next. The idea behind this approach is to harness the entire pixel-value range in the predictions. Figure 4.7 shows the relationship between all different type of predictions.

Model Diff

The architectures used to create the Diff (from difference) can be any of the ones mentioned before. The only thing that changes are the target values, which are the observations with the last context frame subtracted from them. Figure 4.8 schematizes how these targets are obtained, by taking each of them and compute the relative difference with the last frame for the context sequence.

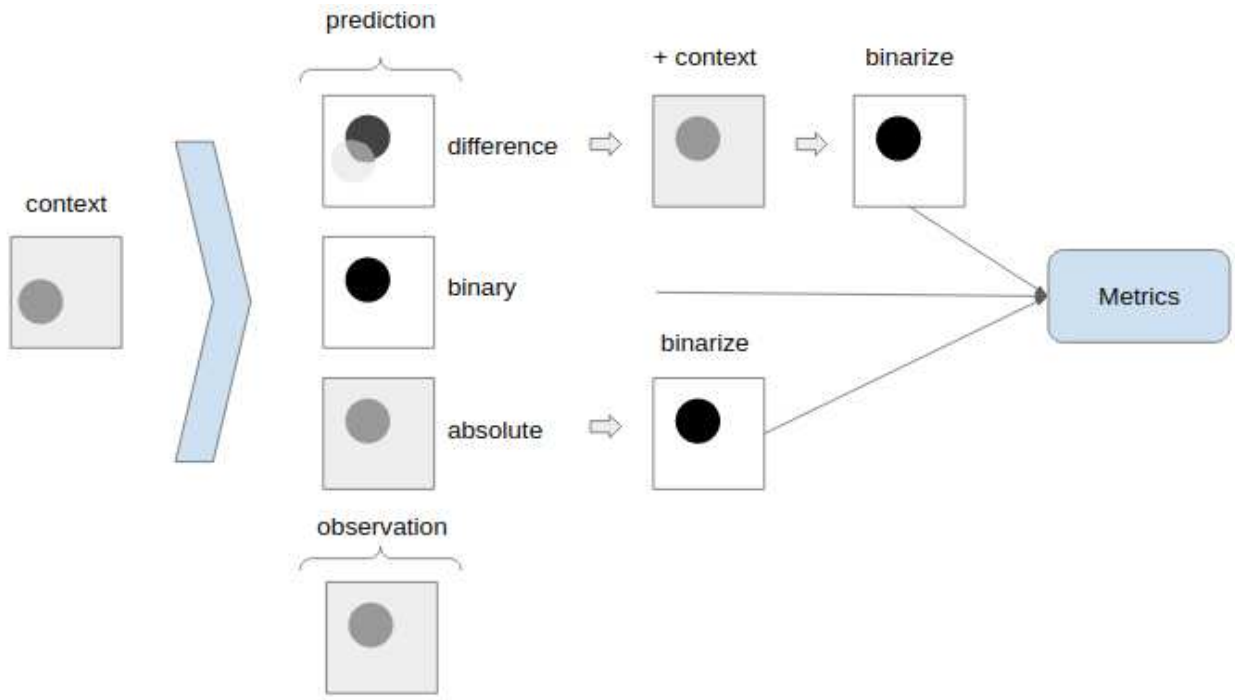


Figure 4.7: Diagram of different possible prediction types. Where 'context' is the history used to make a prediction and the light-blue arrow represents making the prediction.

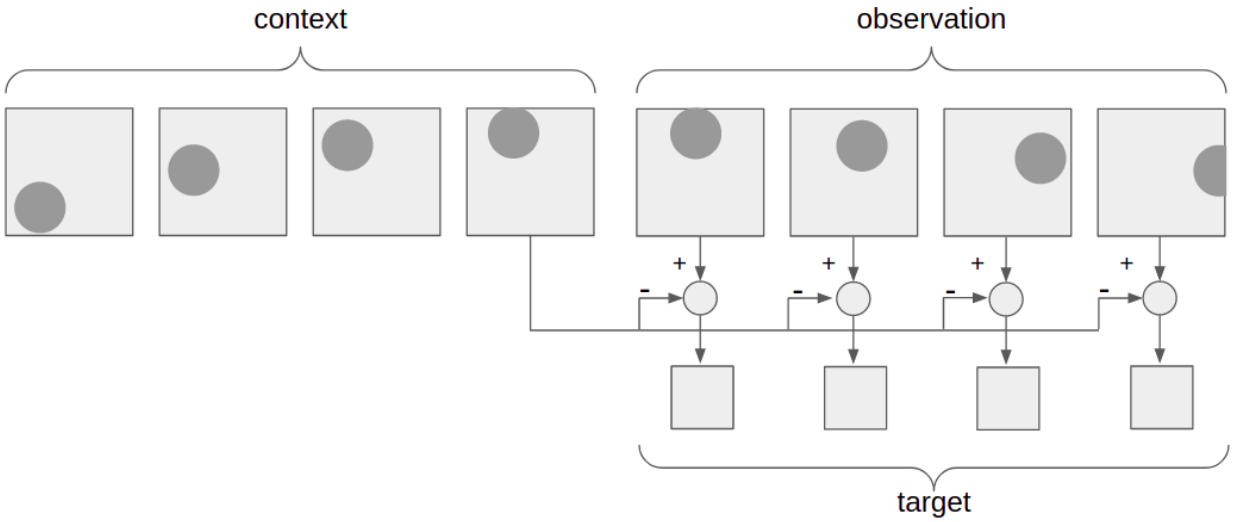


Figure 4.8: Diagram of how the training dataset is modified for the Diff model.

Model Composite

Many of the metrics rely on binary values, while others do not and beyond all the metrics, visual inspection is a key aspect for a meteorologist. Therefore, for the binary approach to be useful a composite model is proposed. It consists of aggregating outputs of different binary models trained

at different reflectivity thresholds scaled to their corresponding reflectivity values. Because it cannot be used an infinite number of binary models to fill all the continuous spectrum of reflectivity values a base prediction of continuous values is added. The goal is to improve the base prediction with all these layers of binary predictions. Figure 4.9 shows a diagram on how this is done and the desired benefit, where if observed, the base prediction is missing some high reflectivity values that in the composite are present thanks to the binary model trained at that reflectivity value.

The architectures used to create the composite can be any of the ones used for the base model. The only thing that changes are the target values, which are binarized using the desired threshold.

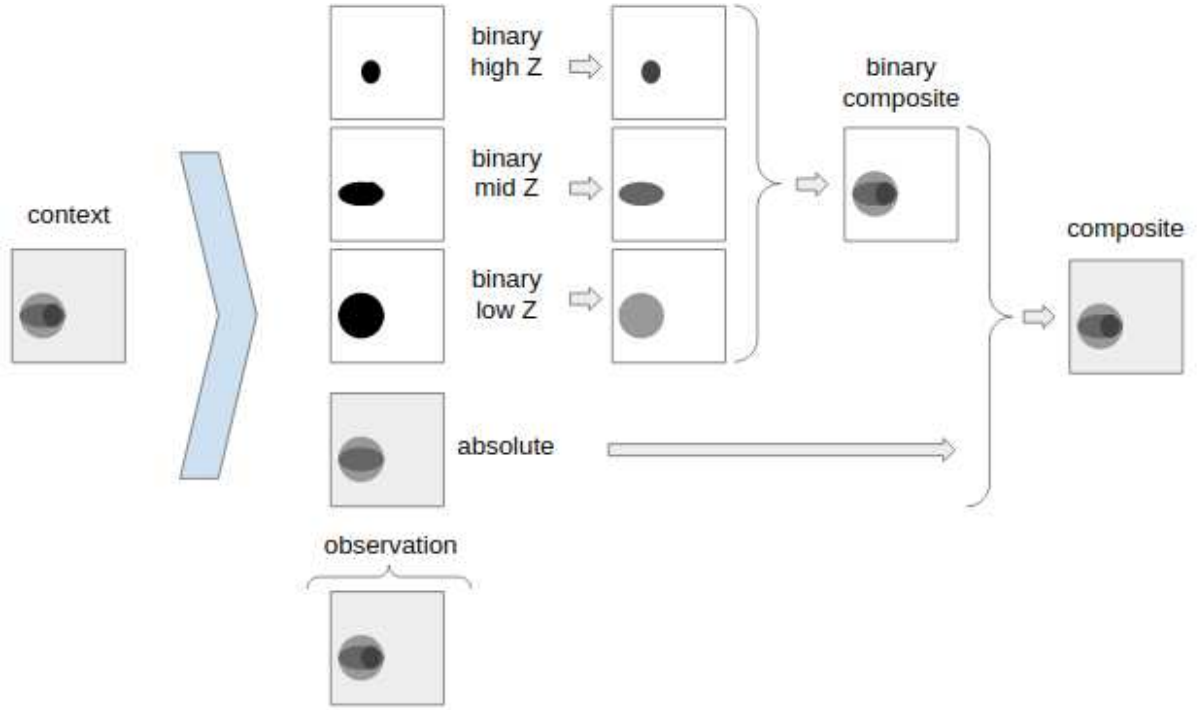


Figure 4.9: Diagram of how the Composite model harness binary predictions.

Finally, the third group of models consists of using different performance loss functions during the training of the models. There were many reasons why to explore all these different functions. In the literature [23, 24] it has been discussed that traditional mean squared error (MSE) yields blurry results. Then, one author in two different papers [38, 40] presented a comparison between MSE and Logcosh showing better results with the latter. Another reason is that in computer vision

it is more common to evaluate the quality of an image using structural similarity metrics such as SSIM and MS-SSIM. As proved in [67] SSIM can be quasi-convex, and therefore it can be used as a loss function [24]. Lastly, in weather nowcasting, higher values of reflectivity are more relevant as they may indicate hazardous scenarios, and they also appear normally in smaller sections of the frames, so a weighted loss function is also tested setting larger weights to higher values [24, 29]. Binary cross-entropy was also used to train the models as they did in [19], [34], and [37]. Lastly, as suggested by [24] and [27], a linear combination of different weighted losses might produce better predictions as they target different aspects of the images. Thus, the 'combined' loss refers to the following formula (taken from [24]) unless specified otherwise:

$$Loss = MSE + 0.1 \cdot MAE + 0.02 \cdot (1 - SSIM) \tag{4.1}$$

each of the individual losses are explained with more details in Section 4.3.

Model MODEL-Loss

For each of the variations with different losses listed in Figure 4.4, they will be referred to as the name of the model hyphen the type of loss used, such as 'ResGRU-SSMI'. For the balanced variation, as it is a modification and not a loss by itself, it will be referred by adding a prefix 'B', such as 'ResGRU-B-MSE'.

4.3 Evaluation metrics

There is no metric that can capture all the aspects of a weather prediction. Because there is not only one intended application for the predictions, for every case a different metric might be better. Thus, it is common for the nowcasting community to use several metrics, and the Developmental Testbed Center has a comprehensive guide [68] that includes most of them. There are two main categories that pertain to this thesis, categorical metrics which use binary values (0 to represent no presence of reflectivity and 1 to represent the corresponding reflectivity value that is being analyzed), and continuous metrics which work directly with continuous reflectivity values.

CATEGORICAL

The categorical metrics are based on the contingency table presented in Table 4.1.

Table 4.1: Contingency table given predictions and observations. T=True, F=False, P=Positive, N=Negative. 1 denotes reflectivity presence and 0 absence.

		Observation	
		1	*0*
Prediction	*1*	TP (hits)	FP (false alarm)
	0	FN (misses)	TN (correct rejection)

Total (T)

$$T = false\,alarm + correct\,rejection + misses + hits \tag{4.2}$$

Total corresponds to the total number of predictions/observations. It is not a metric by itself, but it is used by others. For one frame it corresponds to the count of pixels (frame height times frame width).

Accuracy (ACC)

$$ACC = \frac{hits + correct\,rejection}{T} \tag{4.3}$$

Accuracy is the ratio of correct predictions over the total number of predictions. It ranges from 0 to 1, 1 being the perfect score. The main disadvantage of accuracy is that it considers the correct rejections which are normally much more than the overall positive observations, and the latter are more critical to successfully detect as those indicate high reflectivity intensity.

Probability Of Detection (POD)

$$POD = \frac{hits}{hits + misses} \tag{4.4}$$

Also known as the hit rate, it is the proportion of positive observations correctly predicted. It ranges from 0 to 1, 1 being the perfect score.

False Alarm Ratio (FAR)

$$FAR = \frac{false\,alarm}{false\,alarm + hits} \tag{4.5}$$

It is the proportion of negative observations incorrectly predicted. It ranges from 0 to 1, 0 being the perfect score.

Critical Success Index (CSI)

$$CSI = \frac{hits}{hits + false\,alarm + misses} \tag{4.6}$$

It is the ratio of positive observations correctly predicted over all positive observation and incorrectly predicted negative observations. It ranges from 0 to 1, 1 being the perfect score.

Equitable Threat Score (ETS)

$$C = \frac{(hits + false\,alarm)x(hits + misses)}{T} \tag{4.7}$$

$$ETS = \frac{hits - C}{hits + false\,alarm + misses - C} \tag{4.8}$$

Also known as the Gilbert Skill Score (GSS), it is the similar to Critical Success Index, but it takes into account the chances (C) of having hits by chance. It ranges from $-1/3$ to 1, 1 being the perfect score, and a value of 0 means is not better than a random predictor.

CONTINUOUS

Correlation Coefficient (CORR)

$$CORR = \frac{tr(\bar{\mathbf{P}}\mathbf{O})}{tr(\bar{\mathbf{P}}\mathbf{P})tr(\bar{\mathbf{O}}\mathbf{O})} \tag{4.9}$$

It measures the linear association between the observation (O) and the prediction (P). It is calculated as the ratio of the Frobenius inner product over the Frobenius norm of both matrices $\mathbf{O}$ and $\mathbf{P}$. It ranges from -1 to 1, 1 being the perfect score, and a value of 0 the worst score.

Mean Square Error (MSE)

$$MSE = \frac{1}{T}\sum_{i=1}^{T}(P_i - O_i)^2 \tag{4.10}$$

Mean Absolute Error (MAE)

$$MAE = \frac{1}{T}\sum_{i=1}^{T}|P_i - O_i| \tag{4.11}$$

Unlike MSE, MAE is more robust to outliers. MSE and MAE range from 0 to infinity, 0 being the perfect score.

Structural Similarity (SSIM)

$$SSIM = \frac{(2\mu_P\mu_O + C_1) + (2\sigma_{PO} + C_2)}{(\mu_P^2 + \mu_O^2 + C_1)(\sigma_P^2 + \sigma_O^2 + C_2)} \tag{4.12}$$

where μ is the average, σ the standard deviation, $C_1 = (k_1 L)^2$ and $C_2 = (k_2.L)^2$ stabilizers when the denominator is weak, L the pixel-value dynamic range, and k_1, k_2 constants. The formula above is applied with a sliding window of size 11x11 using a Gaussian weighting function.

It considers the luminance, the contrast, and the structure of the frames. It ranges from -1 to 1, 1 being the perfect score. This metric is not included in [68] as it is not normally used in the nowcasting community. Nevertheless, some authors have used it and it is widely adopted in the

computer vision community. Moreover, it is used also as a loss function for training the models and not only as an evaluation metric.

For this thesis, not all metrics are discussed in the results as many show equivalent results. Nevertheless, they are all presented for any comparison the reader would like to do with other works. The metrics used here are CSI, FAR, POD, MSE, and ETS. Since ETS is the same as CSI but consideris the effect of randomness, and with the datasets used here that randomness coefficient is neglectable, both CSI and ETS are the same.

4.4 Baseline models

To make a more forthcoming evaluation of the proposed methods in this thesis, other nowcasting models are used on the same data to have a direct comparison. The models fall in two main categories, those based on optical flow and the others based on machine learning.

4.4.1 Based on Optical Flow

Extrapolation

It generates a nowcast by applying a simple advection-based extrapolation to the given precipitation field.

S-PROG

The motion field is used to generate a deterministic nowcast with the Spectral Prognosis model, which implements a scale filtering approach in order to progressively remove the unpredictable spatial scales during the forecast.

ANVIL

Originally developed to use with vertically integrated liquid (VIL) data, it can be used with any 2-dimensional input. It is an extrapolation-based nowcast that uses a cascade decomposition and a multiscale autoregressive integrated model that can predict growth and decay.

All the above models are implementations from the pySTEPS project [69], which provides open-source nowcasting models regularly improved. For each model, the estimation of the motion field was done using Lucas-Kanade optical flow method. The hyper-parameters of the models were optimized using Bayesian optimization on the validation dataset.

While extrapolation is a static nowcast, S-PROG can predict growth and decay but with loss of small-scale features and with post-processing to correct bias and wet-area ratios. ANVIL, on the other hand, can predict growth and decay while preserving the small-scale structures without the need for post-processing.

DARTS

Currently deployed in Dallas Forth Worth using CASA radars [12], it uses linear least-squares estimation implemented in the Fourier domain for motion estimation with advection performed via a kernel-based method formulated in the spatial domain.

4.4.2 Based on Machine Learning

RainNet

RainNet [38] is a similar model to those proposed in this thesis using an encoder-decoder architecture and convolutional layers. It was inspired by U-Net [70] and SegNet [71], which are originally designed for binary segmentation of images. This model outperforms the most basic approach of nowcasting which is persistence and the implementation of [40] of a different convolution model.

trajGRU

This is the latest and most promising model implemented by the authors that inspired this thesis [29]. It is based on an encoder-decoder architecture using a modified recurrent neural network with dynamic recurrent connections over time to perform better where location-variant filters are needed. Both, original implementation by the authors in MxNet was used ('trajGRU L17' from

[72]) as well as third-party implementation in PyTorch [73], and only the results from the latter are shown as they were better.

4.5 Training and hyper-parameters tuning

For every model the training was done for 150 to 300 epochs, using Adam optimizer and a step learning scheduler with gamma equal to 0.7 and 3 steps. No variations on the optimizer were tested, except for a simple test using Stochastic Gradient Descent (SGD) which yielded considerably worst results.

The optimization of the hyper-parameters was done first by hand, narrowing the range, and then using Bayesian optimization [74] to fine-tune them. The list of such parameters is presented in Table 4.2.

Table 4.2: Hyper-parameters and the tuning ranges. 'Choice' means one of the listed values, 'uniform' means a pick for a uniform distribution between the lower and upper ranges listed, and 'log' similar but from a logarithmic distribution.

Parameter	Ranges	Description
weight decay	choice{0,0.01,0.1}	Adam parameter
beta1	uniform{0.5,0.9}	Adam parameter
beta2	uniform{0.9,1}	Adam parameter
learning rate	log{1E-2,1E-5}	Learning rate optimizer parameter
filters	choice{64,96,128,256}	Base number for convolutional filters
dropout	uniform{0,0.5}	Dropout rate

The final values used can be found on the code repository [66].

4.6 Environment and Software

The training was done using a single GPU GeForce GTX TITAN X with 12 GB of memory. Some of the most relevant software packages with their versions are listed in Table 4.3.

Table 4.3: Software used in this thesis.

Software	Version	Description
CUDA	10.1	Computing platform for GPUs
Conda	4.8.2	Python packages platform
Python	3.7.3	Programming language
PyTorch	1.3.1	Machine learning library
Keras	2.2.4	High-level API for Machine learning
Tensorflow-gpu	1.14.0	Machine learning library
PySteps	5.0.1	Package for short-term weather predictions
Cartopy	0.18.0	Package for geospatial data processing
HyperOpt	0.2.3	Package for Bayesian hyperparameter optimization
Innvestigate	1.0.8	Package for NN analysis
MXNet	1.5.1	Machine learning library
Pytorch-SSIM	0.1.5	Package for SSIM calculation in torch tensors

4.7 Experiments

The experiments are organized as follow. First, there are two experiments related to training phase, one analyzing which is the best possible checkpoint to save during the training, and the other one which is the best loss function to use as the performance metric of the optimizer. The second part consist of the experiment using different types of predictions, being those introduce in Section 4.2. Then, three experiments are proposed on how the dataset can be best used. This refers to the length and sampling rate on the history and prediction sequences. Afterwards, the comparison against baseline models is done. Finally, it is analysis how varying the geographical location of the data can affect the performance of the proposed models.

4.7.1 Behavior over epochs

As there is no unique metric to capture all aspects determining how well the predictions are, the training must be done only using one or certain combinations of a few. Therefore, the question of whether the training should be stopped based on the performance metric or other metric, which can be non-convex, is raised. To test this, both the best performance metric checkpoint and the best external metric checkpoint are saved and compared. The last checkpoint (given an arbitrary number

of training epochs) is also saved and compared. As exemplified in Figure 4.10, the validation score starts to deteriorate due to overfitting, using this last checkpoint allows to test whether the overfitting occurs only on that metric or in general.

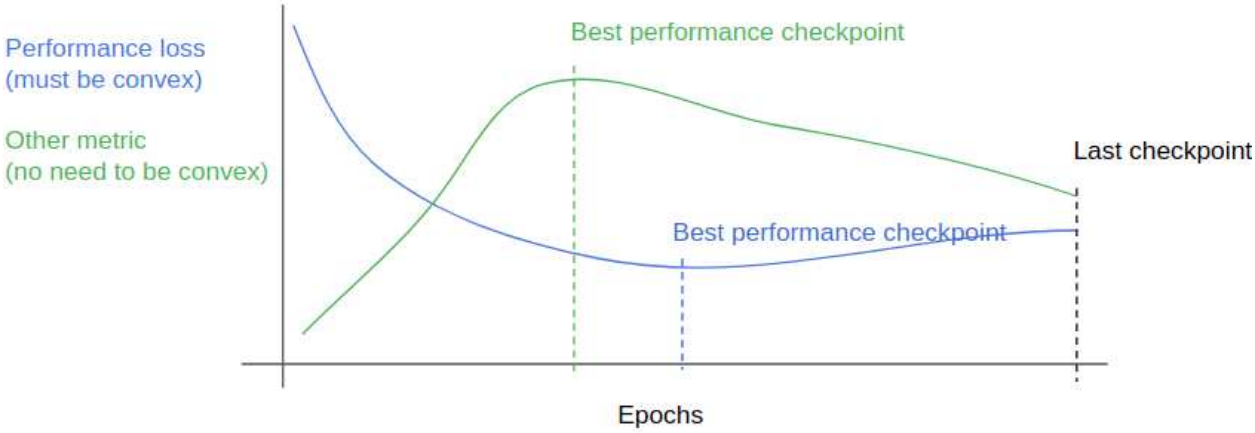


Figure 4.10: Example of a training plot for the experiment saving checkpoints based on different criteria.

4.7.2 Comparing different losses

The intention here is to replicate some of the results presented here [24] when comparing the performance of the same model trained with different losses.

- Mean Square Error (MSE)
- Mean Absolute Error (MAE)
- Balanced-Mean Square Error (B-MSE)
- Binary Cross-Entropy (BCE)
- Structural Similarity (SSIM)
- Logarithm of the Hyperbolic Cosine (LogCosH)
- Linear combination of the above as in Eq. 4.1 (Combined)

4.7.3 Comparing different prediction types

In this experiment, the goal is to analyze if predicting the absolute values of the frames is better than predicting the difference, and if targeting a specific reflectivity range by using a binary output

is better than aiming to capture the whole frame. For details refer back to Section 4.2 where these two models are explained.

4.7.4 Behavior on different history and prediction lengths

Most of the literature reviewed in Chapter 3 made predictions of 10 frames or less, meaning that the lead time goes only up to at most 1 hour, using the median sampling rate of 6 minutes. Moreover, none of the previous works analyzed how the predictions differ when a different number of frames are used to make the predictions or when the number of predicted frames varies. Therefore, in this experiment, the variations on the lengths of both the history (context) and predictions are studied.

In order to do this, two different methods are used. First, using the same dataset of 64 frames per event, three different architectures are created. One to predict 8 frames given 8 frames, and the other two similar but with 16 and 32 frames respectively. In Figure 4.11 (a) a scheme of how the frames from the dataset are being used in this experiment are shown. By comparing these models, the intention is to observe how the predictions' scores are on the overlapping frames (i.e. the first 8 predictions), and then how the metrics deteriorate in the last frames.

The second part of the experiment consists of keeping constant the predicted frames to 16 and varying the history lengths, using 8, 16, and 32 frames. Figure 4.11 (b) shows a scheme of how the frames from the dataset are being used in this part of the experiment. The objective of this experiment is to test whether increasing the lead time in the predictions would cause an improvement overall the predictions as more detail might be necessary to extend the forecast further in time, or would it be counterproductive as more information needs to be remembered and quality will be sacrificed.

The dataset used here (Training Mix from Figure 4.2a) differs from the one being used so far (Training DFW) in the number of frames per event. Also, the handling of the dataset differs as in previous experiments all frames were used during the training, but here the frames are fixed to make a valid comparison. The diagram in Figure 4.12 exemplifies how this was done. In case

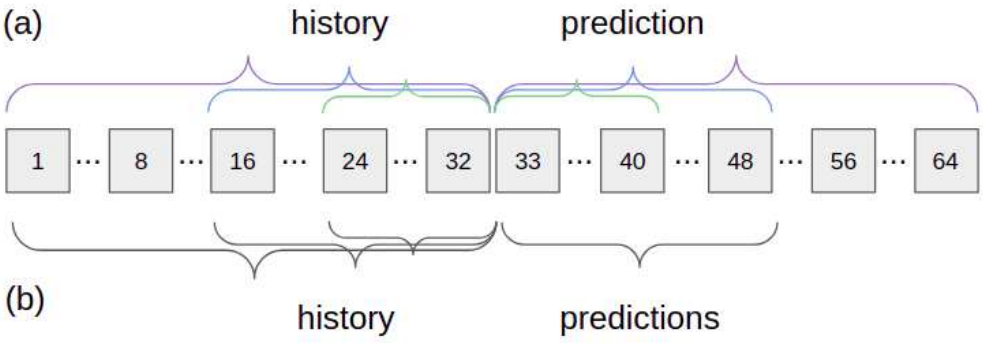


Figure 4.11: Example of experiments (a) varying lengths to predict more frames, and (b) varying lengths of history while keeping constant the number of predicted frames.

(a), where the starting frame is not fixed on each epoch the history and observation (ground truth) frames can shift along with the whole event. So, if for example, the training dataset has events of 40 frames each and the model being trained takes 16 for both history and observation there are 25 possible starting points. On the other hand, in case (b) where the boundary frame between history and observation frames is fixed, in every epoch, the selected frames are the same. The reason to make the training deterministic is to keep constant the number of seen events during training when two models with different history/observation frames are trained. Moreover, the purpose to fix the middle frame and not the starting frame is to minimize variance on the seen frames for the before mentioned scenario. A consequence of making it deterministic is that the net amount of samples for training is reduced to the number of events in the dataset, while with the random approach there is a multiplication factor on this number (25 for the example used above). Therefore, to have more samples, the 'mix' dataset (from Figure 4.2), which contains samples from multiple location, is used.

4.7.5 Behavior on different prediction steps

In the same line as the previous experiment, the intention here is to test how the model behaves at different time resolutions. For that, the same model is trained with all the frames in each event and compared to the model trained on every other frame in each event. Then, another model using double the number of frames for the history and predictions is also added to the comparison to

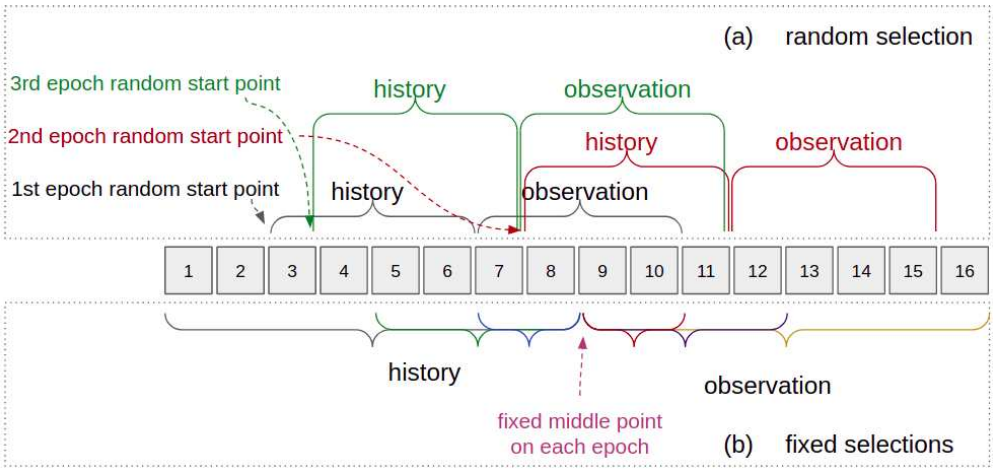


Figure 4.12: Example of (a) using a random part, or (b) a fixed part of the dataset for the training.

mitigate the possibility that the predictions changed due to the lengths of the history used. All these variation in the portion of the dataset used are schematized in Figure 4.13. Nonetheless, this should be considered as a guideline as having two variables changing (step size and number of frames) is not possible to draw a definitive conclusion on the results.

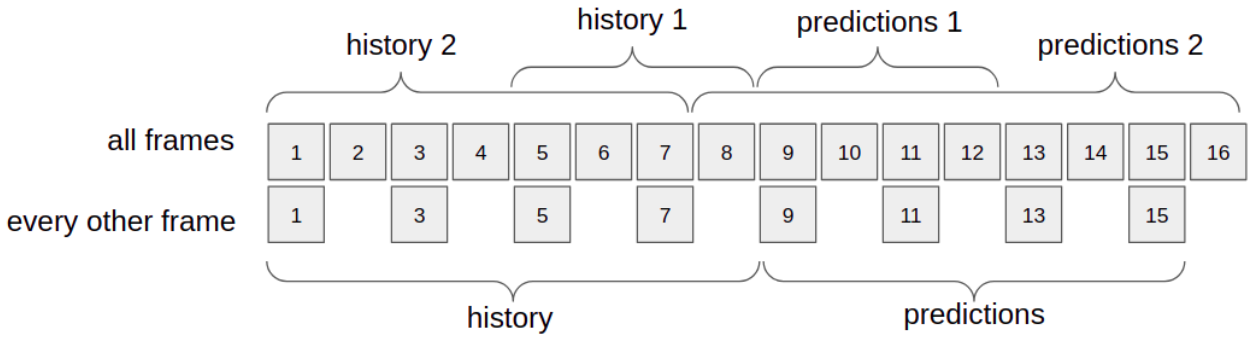


Figure 4.13: Example of varying-step experiment.

4.7.6 Comparing proposed models against baseline models

In this experiment different models, named ResGRU, ResConv, and Composite (using ResGRU as base), are compared between each other and against the baseline models listed in Section 4.4. Because of the nature of each baseline model, the experiment is split into two parts. First, the three models proposed here and the baseline models, DARTS, trajGRU, and RainNet, are compared

using the non-regularized dataset. This decision was made because DARTS performed better on the non-regularized dataset, and despite the ML models doing better on the regularized dataset the intention here is to provide the most meaningful comparison so the decision was in benefit of the baseline.

The second part consists of comparing ResConv against the PySteps models. In this case, the regularized dataset is used. Another important consideration here is that the predictions of these models cover only a section of the frame so the metric for them is computed only at the valid sections.

The comparison against DARTS is one of the most relevant experiments in this thesis as it compares the proposed model against a real-world nowcasting system, and the nowcasts were done externally. On the other hand, despite using PySteps, trajGRU, and RainNet has the same intentions, the nowcasts were locally done. PySpteps models were optimized on the validation dataset and the ML models were trained on the training set. Regardless of the intentions to produce the best nowcast out of them, it is likely that the original authors can further optimize their models for the datasets used in this thesis and obtained better results.

4.7.7 Performance on different geographical regions

As machine learning models are heavily dependent on the training set, it is of interest to compare the performance of the model on a dataset from a different location. For this, the Denver Spring 2017 dataset is used as well as the Dallas Forth Worth Spring 2019 dataset (same radar as the training dataset). The ideal comparison would be doing a cross-training/testing with the same model trained on radar A and tested on radar A and B, and then trained on radar B and tested on radar A and B. But, because of the lack of availability on two different training datasets a different approach is taken. The comparison is done with DARTS nowcasting system, which is independent (in the sense that no training data is used) of the radar. Therefore, the analysis is done on how relatively similar the nowcast are between DARTS and ResConv on one dataset and then in another dataset.

Chapter 5

Results

5.1 Behavior over epochs

For this experiment, the results using the ResConv with the LogCosh loss are shown.

In Figure 5.1 it can be seen how the model starts to overfit at epoch 28 where the validation loss is at the minimum, but the training loss keeps decreasing. On the other hand, the SSIM, which is just a reference metric (meaning that it is not involved in any computation during the training phase) keeps increasing a bit longer for the validation data, reaching its highest peak at epoch 70. Finally, the training was continued for a total of 300 epochs.

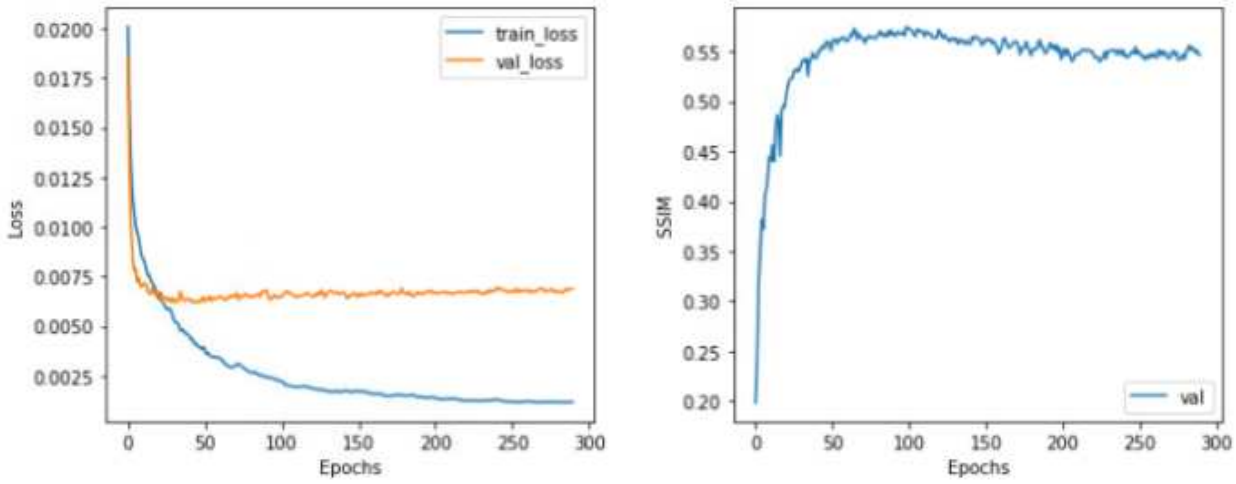


Figure 5.1: Training plots for ResConv-LogCosh.

Figure 5.2 shows the prediction for one event of the testing dataset. The first row corresponds to the observations (ground truth) and the following rows to predictions using different checkpoints along the training. Looking at these predictions it is hard to identify the best one despite they do present differences (some marked with circles). Thus, the metrics in Figure 5.3 do not represent only one event as the shown predictions but it is an average over the testing dataset. The checkpoint

when the performance loss is the lowest (blue) has the best performance in many of the metrics and it does not perform worst in any (considering every time step and standard deviation overlapping).

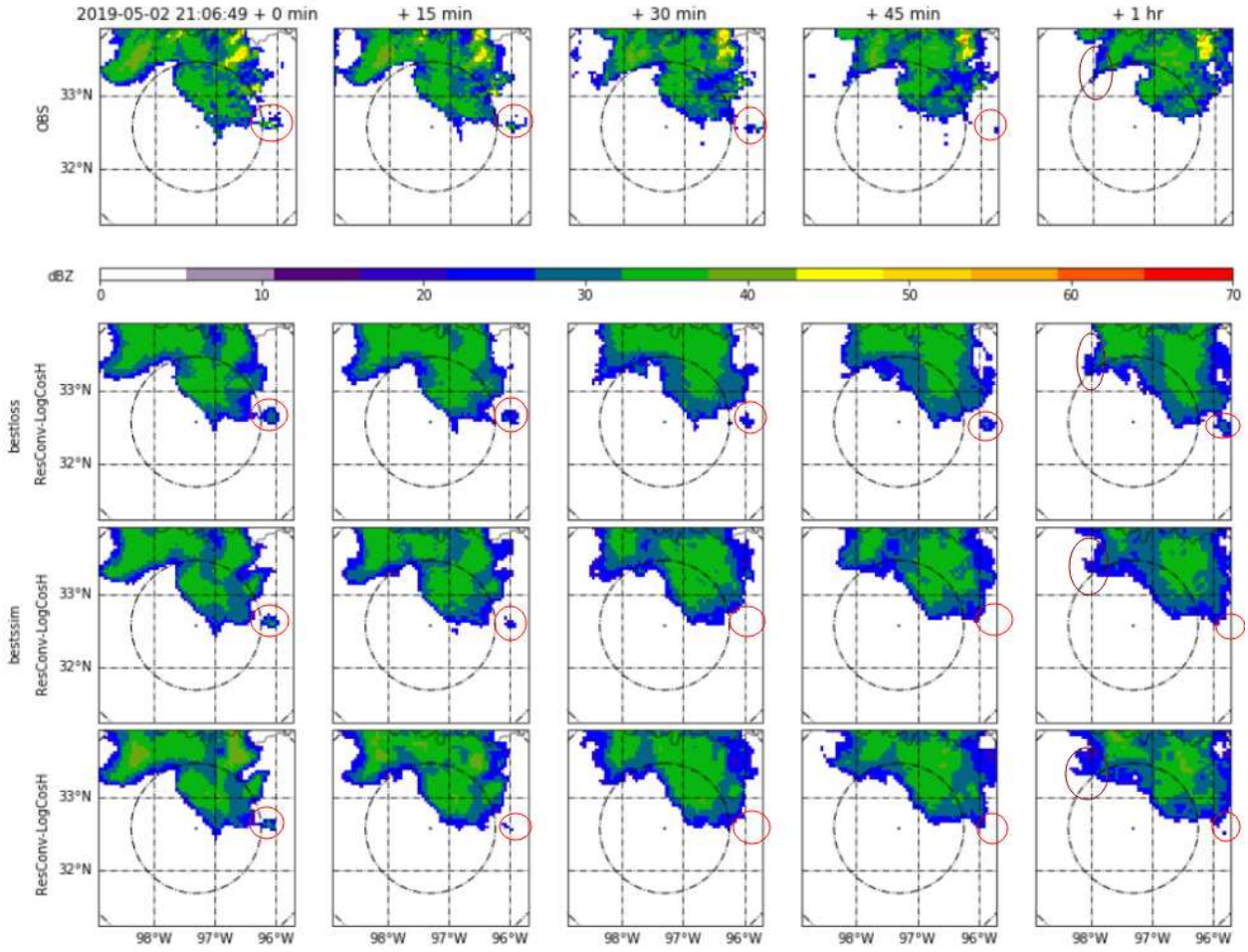


Figure 5.2: Nowcasting using ResConv-LogCosh at different checkpoints. 'bestloss' is at the lowest Log-Cosh, 'bestssim' is at the highest SSIM, both over the validation dataset. And the last row is at the end of the training.

This behavior on the training plots was repeated over most of the different losses and architectures. The exception was on the binary models, where the SSIM kept increasing (Figure 5.4) over the epochs but the metrics did not show any improvement compared to using the checkpoint at the minimum of the performance metric (Figure 5.5).

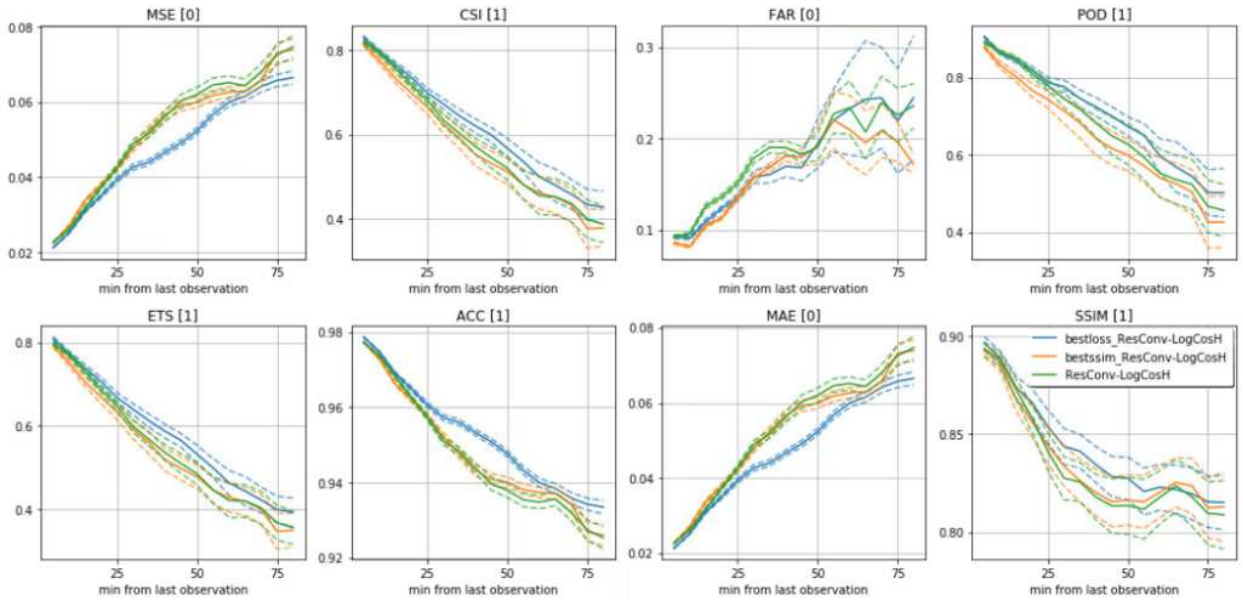


Figure 5.3: Metric plots at different checkpoints.

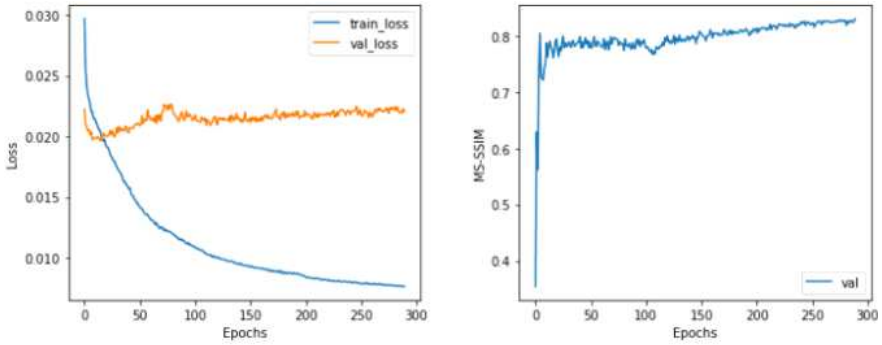


Figure 5.4: Training plots for ResConv binary at 35 dBZ using MSE as the performance metric.

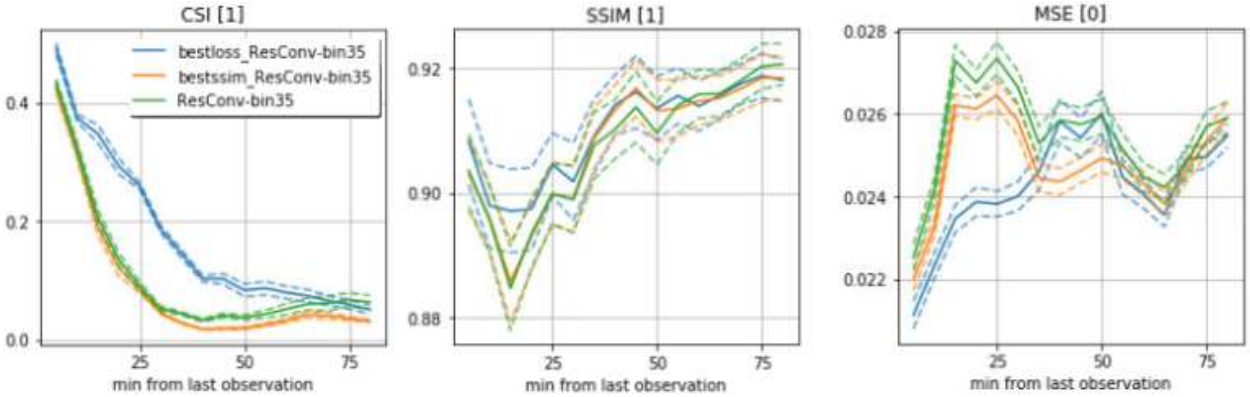


Figure 5.5: Metric plots at different checkpoints a the binary model.

5.2 Comparing different losses

The results presented below are using the same architecture ResConv trained with different performance metrics. There is no clear best metric choice, as some predicted better higher values or frames further away in time, while others performed better closer in time or with lower Z values. Figure 5.6 shows the CSI score for four different thresholds and the behavior of each model varies significantly. For example, using the Combined loss seems to have good results at 25 dBZ in the middle frames and the best in the last frame, but at 30 dBZ is the one performing the worst in almost every frame. MSE and Balanced-MSE are the only ones maintaining a similar behavior over all thresholds, while LogCosh, MAE, BCE, and SSIM did as well but at 35 dBZ the performance deteriorates significantly.

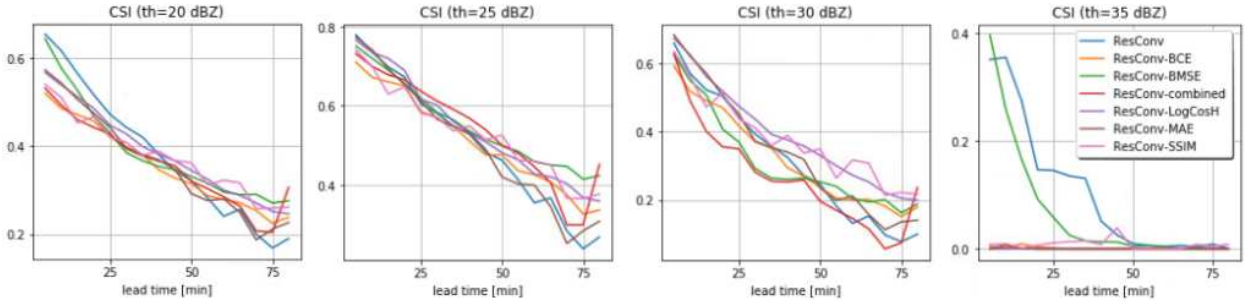


Figure 5.6: CSI plots corresponding at different reflectivity thresholds for different performance losses.

To be able to make a more straightforward comparison the average over multiple thresholds is presented in Figure 5.7. The first plot on the left shows the MSE score where the Combined, LogCosH, MSE are the models performing the best. At analyzing the CSI score the MSE model has the best score at closest frames and the worst at the furthest frames in time. This may suggest that the blurriness effect discussed in Chapter 3 is impairing the predictions. The Balanced-MSE model has good CSI scores but at the expense of having poor FAR meaning that it overestimates reflectivity values. It seems that LogCosH has consistent behavior over all scores.

The predictions in Figure 5.8 exemplify how the different performance metrics affect the predictions. In the last frame from the observation, 3 areas are marked in yellow, orange, and red

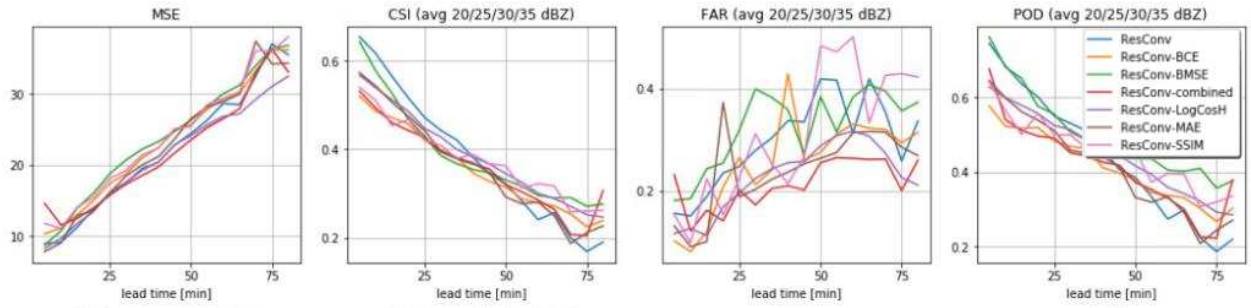


Figure 5.7: Average of metrics corresponding to different performance losses over different reflectivity thresholds.

where many of the predictions highly differ. Only the Combined and MAE models were close to correctly predict the red area. None did a correct prediction on the orange area, but the LogCosH model was the best. For the yellow area, Combined and LogCosH models did well. Unfortunately, the Combined model has a really poor performance on high reflectively values as it did not predict any value above 30 dBZ.

In the Appendix (Figure A.1 and Figure A.2), two more examples using different linear combinations of metrics are presented.

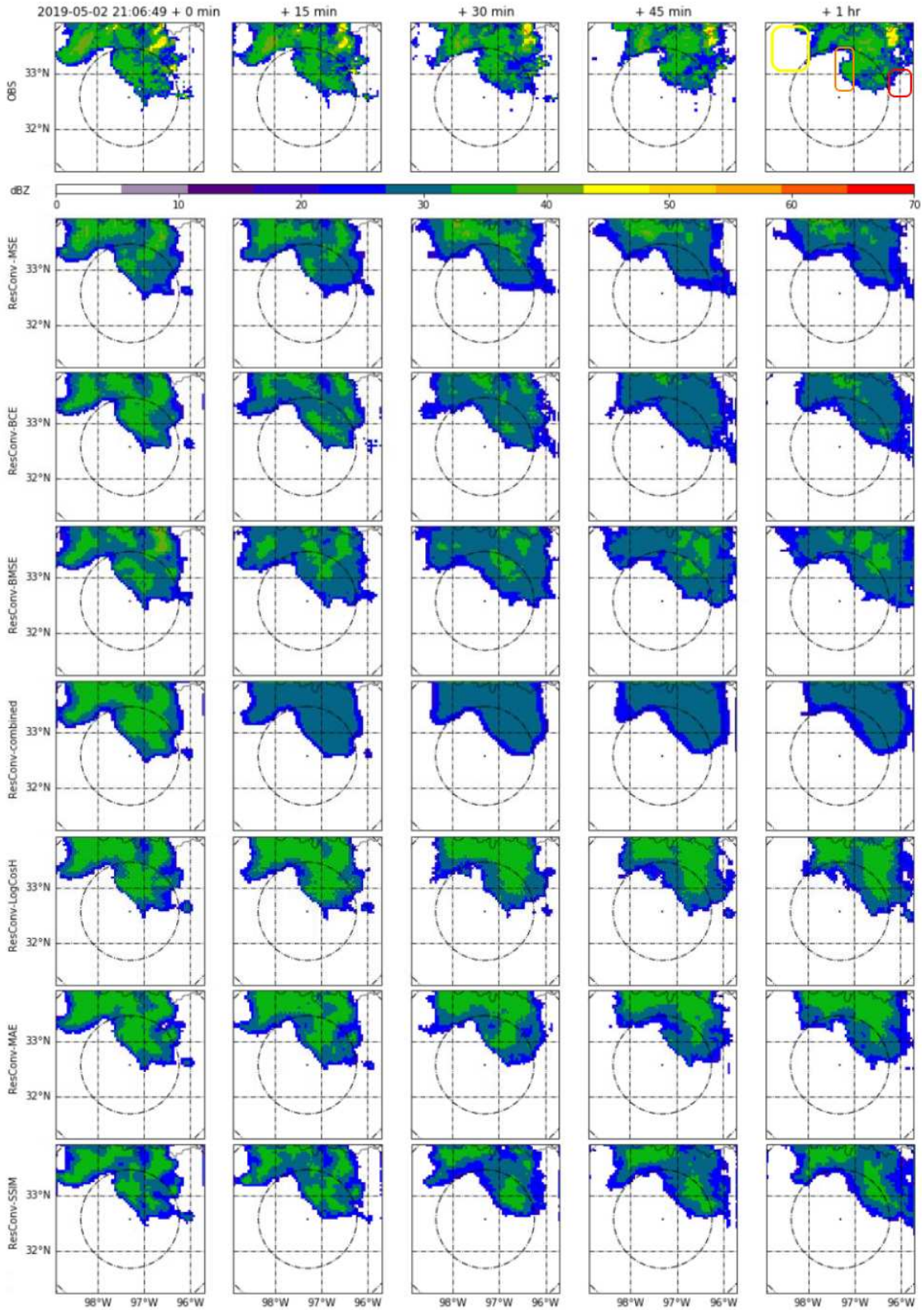


Figure 5.8: Training plots for ResConv trained on different performance metrics.

5.3 Comparing different prediction types

The first experiment shows how using a model trained to predict a specific threshold value can outperform a general model targeting all dBZ ranges. Figure 5.9 exemplifies how at lower reflectivity values predictions do not show too many differences, but at higher Z where the general model fails the specific model still is able to predict values. In Figure 5.10 the metrics over all events show that the specific model for a certain threshold does not outperform the general model at every time step or threshold. It seems that the benefit of the former is mainly at higher Z values for long time predictions.

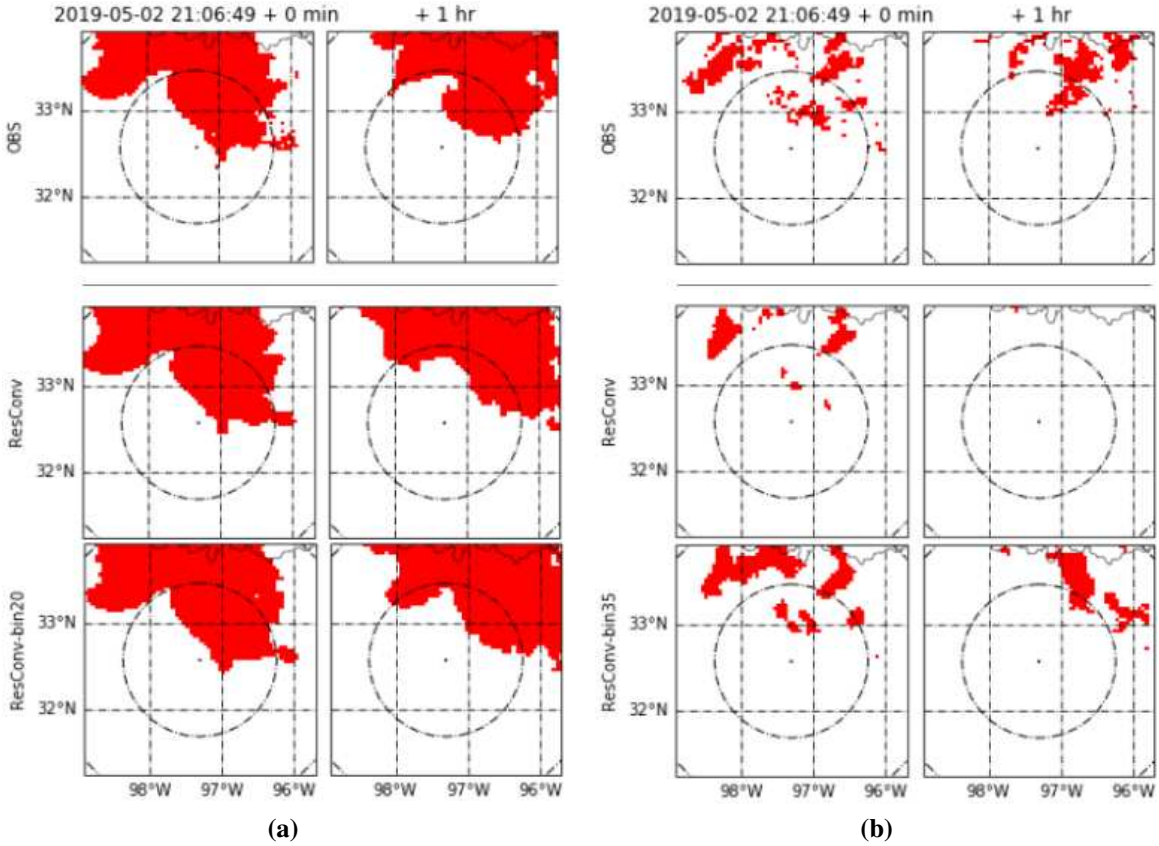


Figure 5.9: Predictions at different thresholds,(a) at 20 dBZ and (b) at 35 dBZ, using a general model ResConv and a model targeting that threshold ResConv-binXX (where XX is the threshold in dBZ).

For the second experiment, a comparison between predicting the absolute values of each frame and predicting the relative changes to the last observed frame is done. Using the latter approach

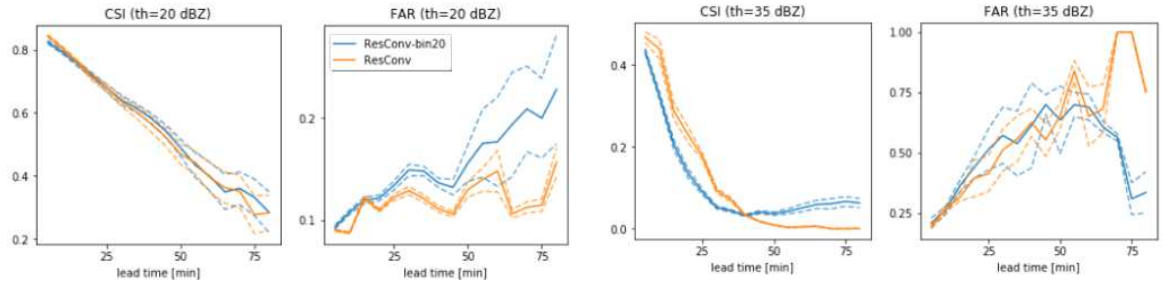


Figure 5.10: Metrics corresponding to experiment of Figure 5.9 average over all testing dataset.

did not produce good results. The metrics were worse on average (over several thresholds) than predicting the absolute values as can be seen in Figure 5.11. Also, looking at the example in Figure 5.12 two things can be noticed. On one hand, the advection is poorly predicted. On the other hand, the high reflectivity values are not being 'lost' along time as it does when using the absolute-value approach. The models for predicting the relative values have not been optimized for that but are the exact same models used for absolute value. Therefore, maybe it is possible to harness the benefit of the relative prediction on high Z values by further optimizing the model.

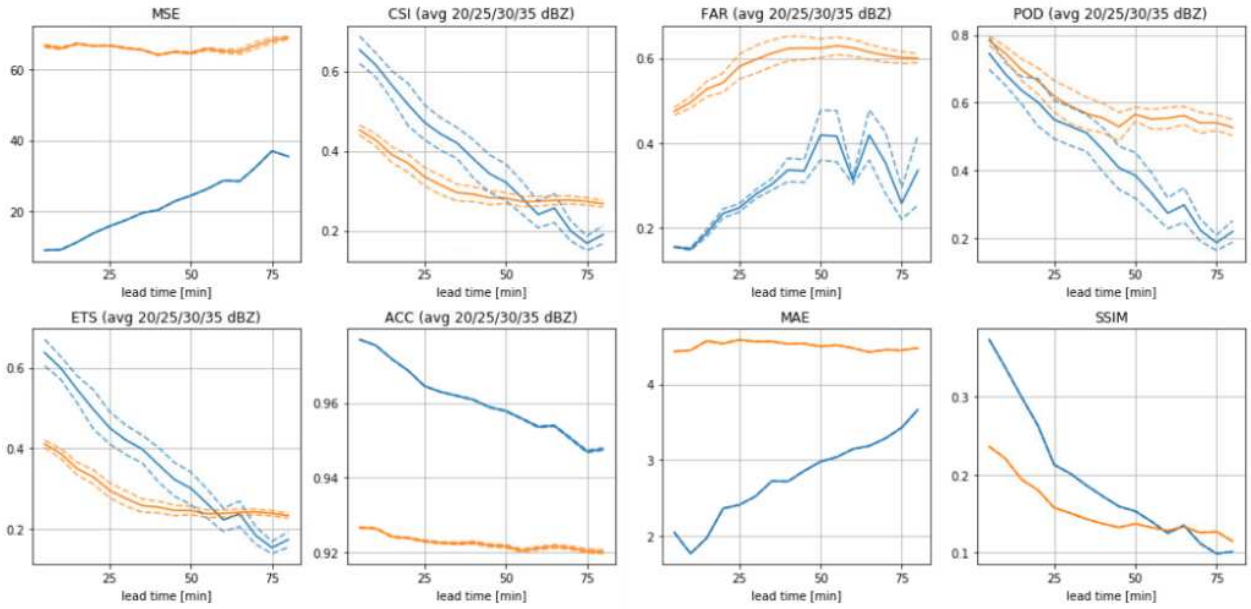


Figure 5.11: Average of metrics at different reflectivity thresholds comparing relative prediction (ResConv-diff) against absolute predictions (ResConv).

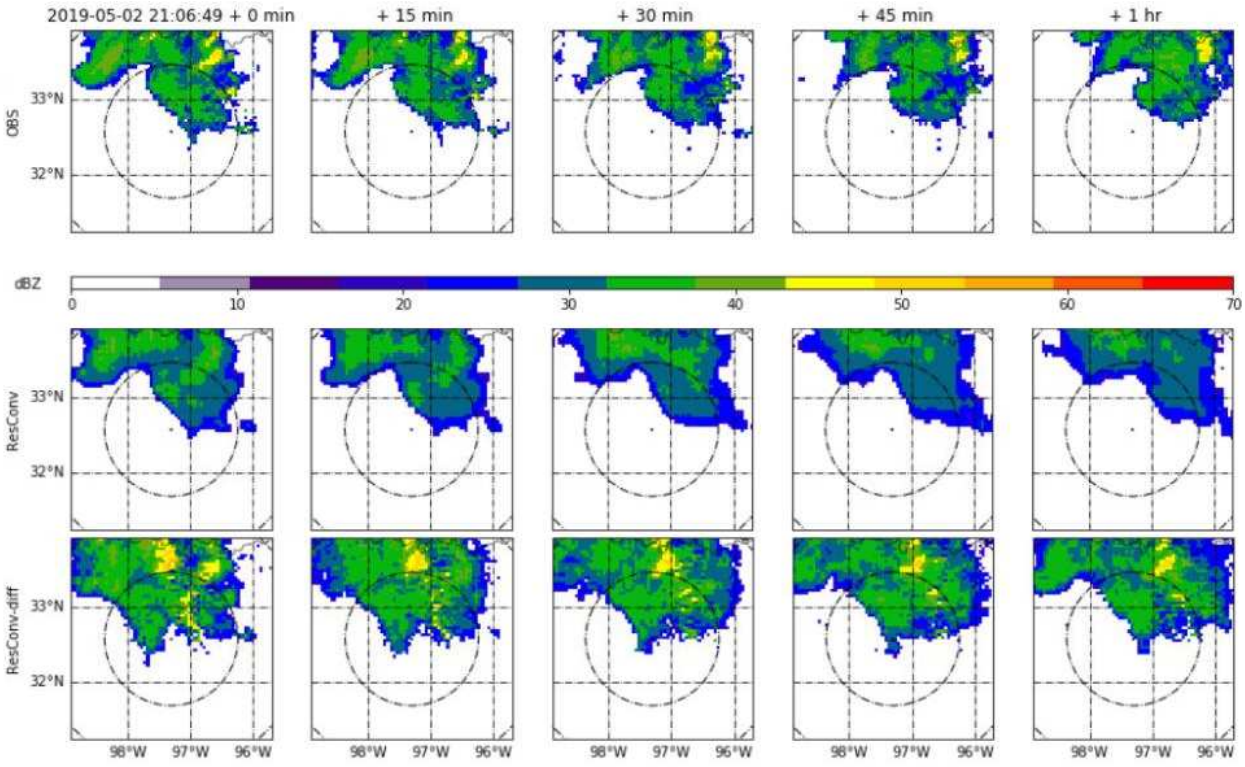


Figure 5.12: Prediction plots for ResConv trained on absolute and relative values.

5.4 Creating a composite prediction

In this section, an example of the composite model, introduced in Section 4.2, is presented. The composite model consists of stacking different models' predictions, as explained in Section 4.2. First, it uses a base model (ResConv in this example) to provide continuous values in the whole reflectivity range. Then, seven binary layers are stacked by keeping the maximum value, each trained to predict a binary output using different reflectivity values, ranging from 20 dBZ to 50 dBZ. Above, 45 dBZ no values are predicted so those layers are not shown here.

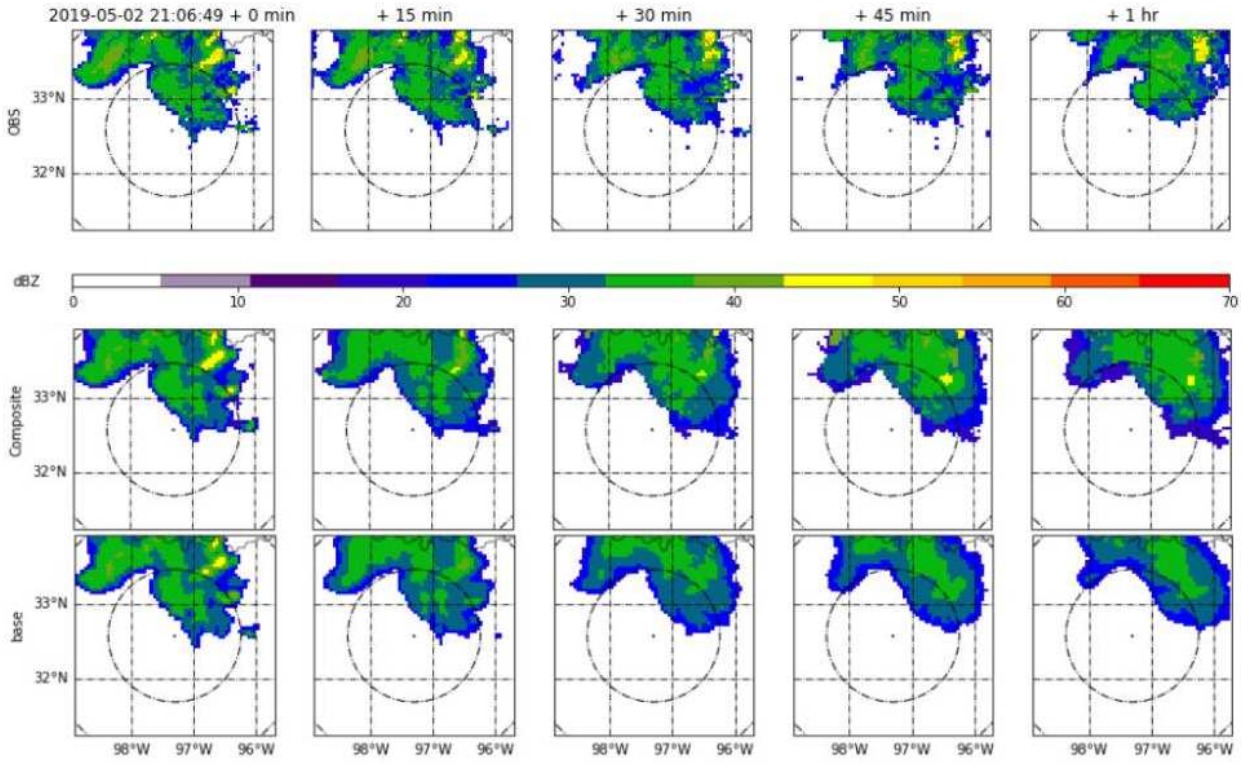


Figure 5.13: Prediction plots comparing the observation with the base model and the composite model.

Figure 5.13 shows how the base model differs from the composite model. The first thing to be noticed is the presence of higher reflectivity values in the latter model. Then, the initial frames of both models are quite similar presenting differences above the eighth frame (30 minutes lead time). When comparing the scores, the composite model did better at every threshold and the average is shown in Figure 5.14. The only metric that it was outperformed by the base model was

the False Alarm Rate (FAR), which normally happens when the nowcasting is overestimating. A clear example is the reflectivity around 45 dBZ (yellow) where it can be observed that the location at later frames is not correct, increasing the FAR value and decreasing the scores of POD and CSI.

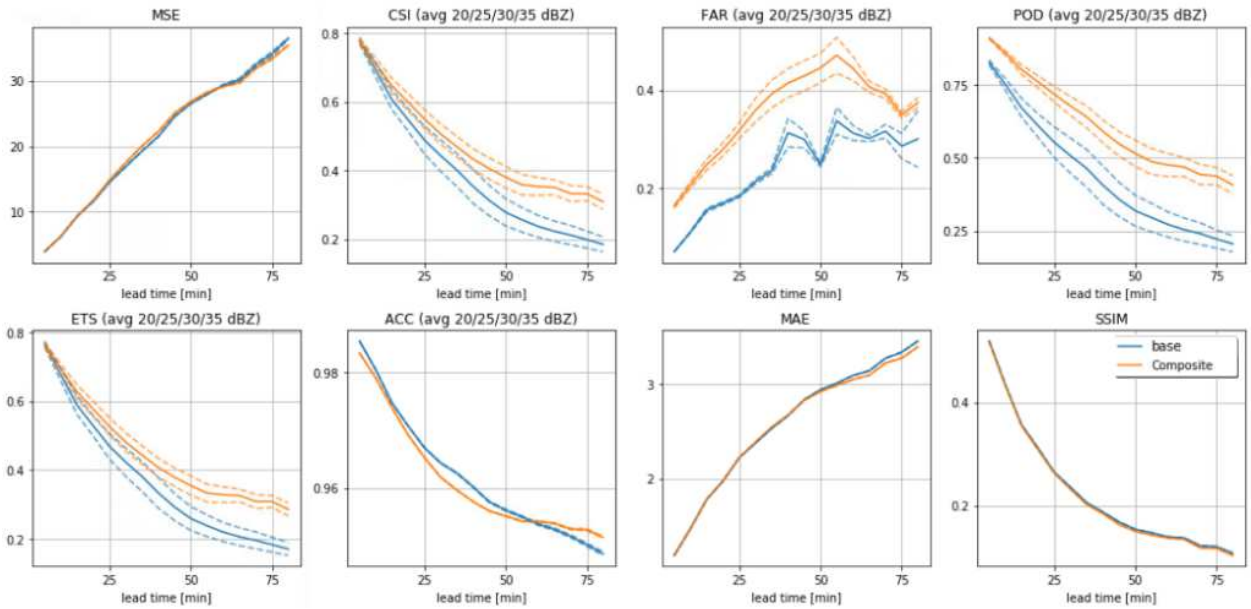


Figure 5.14: Average of metrics at different reflectivity thresholds comparing the base model with the composite.

To get more detailed on how each layer contributes to the model, Figure 5.15 shows each individually at their respective threshold.

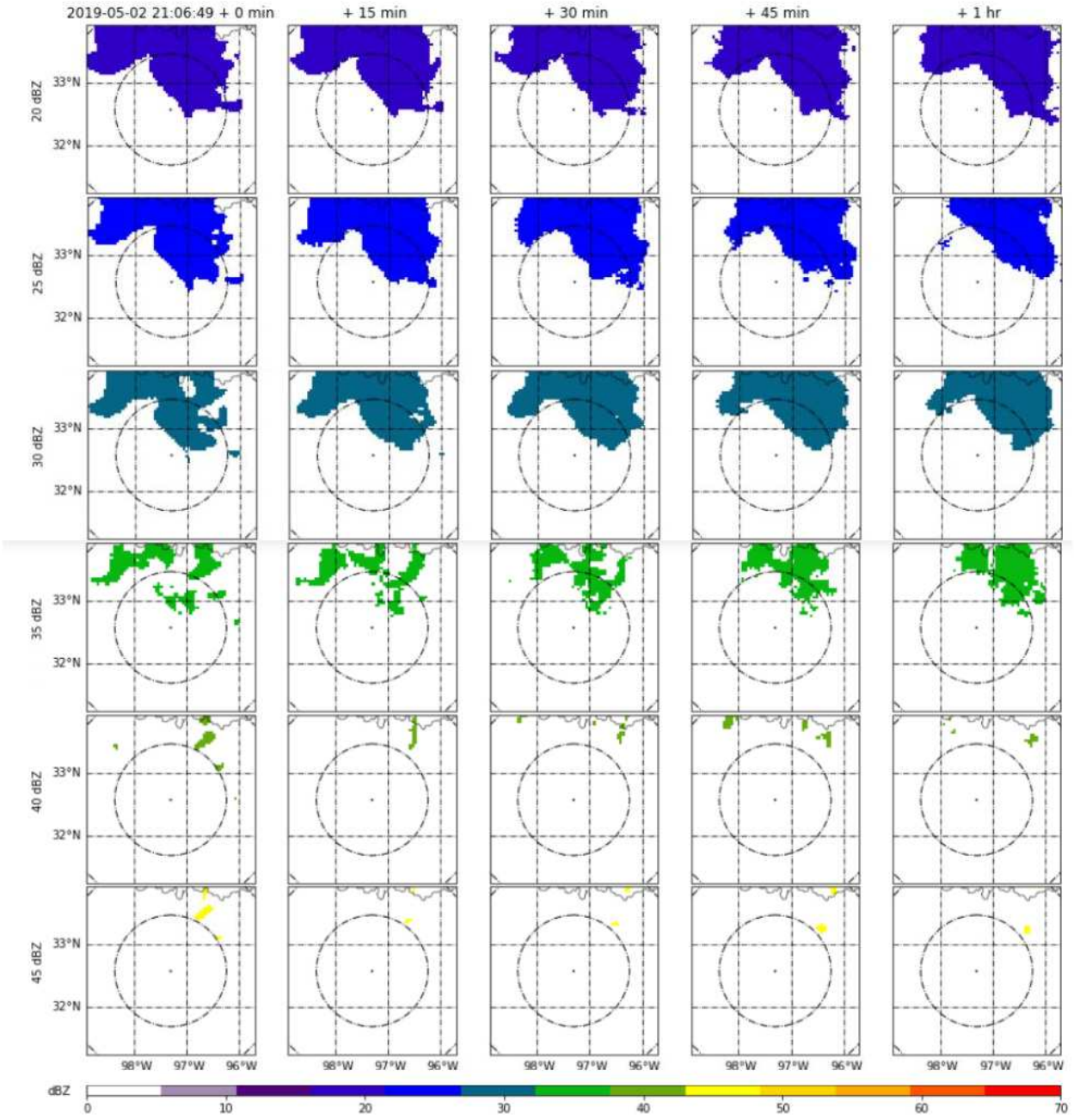


Figure 5.15: Prediction plots of each binary layer in the composite model.

5.5 Behavior on different history and prediction lengths

The following results are from the experiment on varying both the length of the history used to make the predictions and the number of predicted frames. Figure 5.16 shows the metrics when the furthest lead time was left constant (16 frames) but the history used to predict varied from 8, 16, and 32 frames. In general, there were not too many differences between the models' predictions. As observed in the second row of metrics, they correspond for each threshold while the first row is the average of all the thresholds, and the only significant difference is at 35 dBZ.

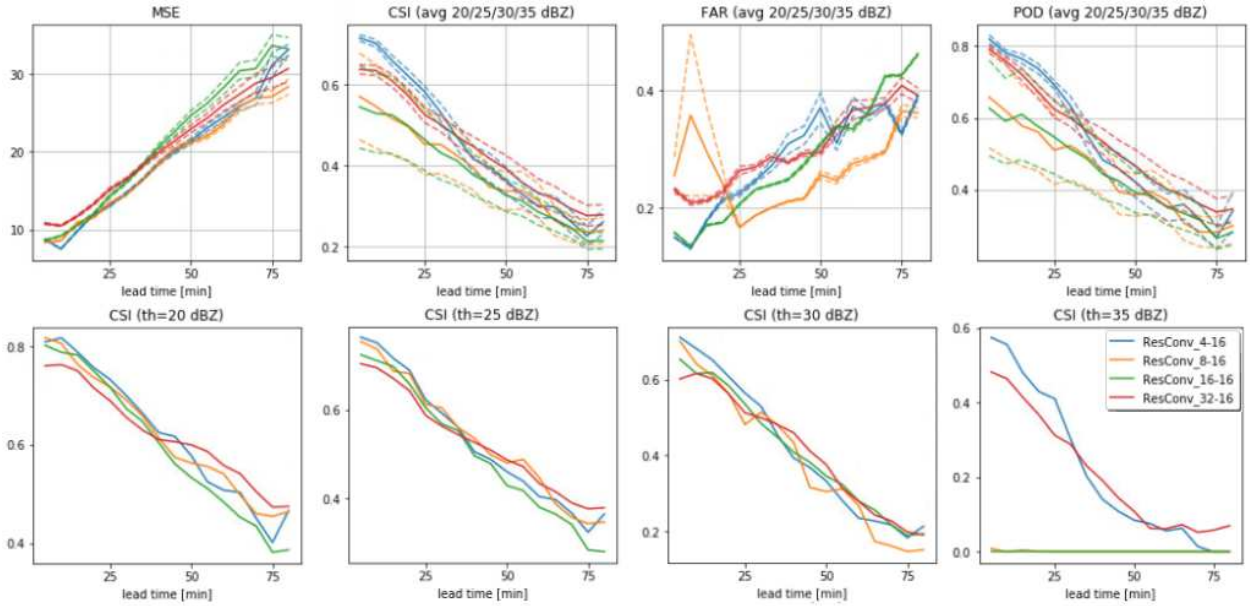


Figure 5.16: Metrics of different models varying the history lengths.

An example prediction is analyzed in Figure 5.17, the similarity between them can also be visualized. Nevertheless, there are three differences, two marked with circles and the third on the shape and size of the light green component (35 dBZ) in the last frame. The red circle marks a gap with low reflectivity that was partially predicted only by the ResConv_4-16 and ResConv_16-16 models.

The next part consists of varying the number of predicted frames while the history observed remains constant. In this case, as the metrics are computed over the predictions and they have

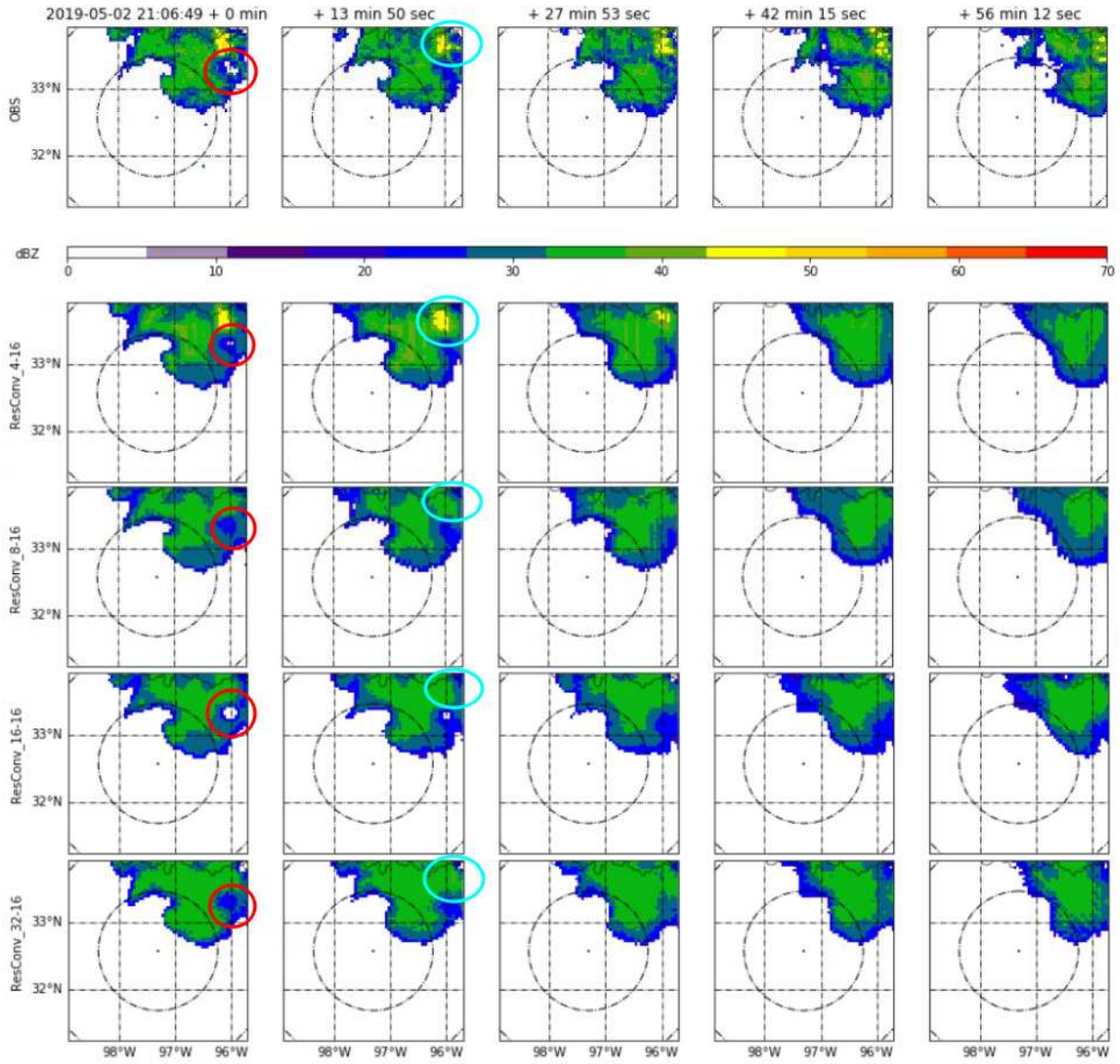


Figure 5.17: Prediction plots of models with different history lengths.

different lengths; the comparison is done only where they overlap. Figure 5.18 shows the previous metrics, and Figure 5.19 displays an example prediction.

The results are clear both on the metrics and on the example predictions. The longer the lead time the model is trying to predict, the more affected intermediate frames are. Nevertheless, looking at the CSI scores, if an extrapolation would have to be made all three models converge to a similar value towards the furthest lead time. This means that there is an improvement in using

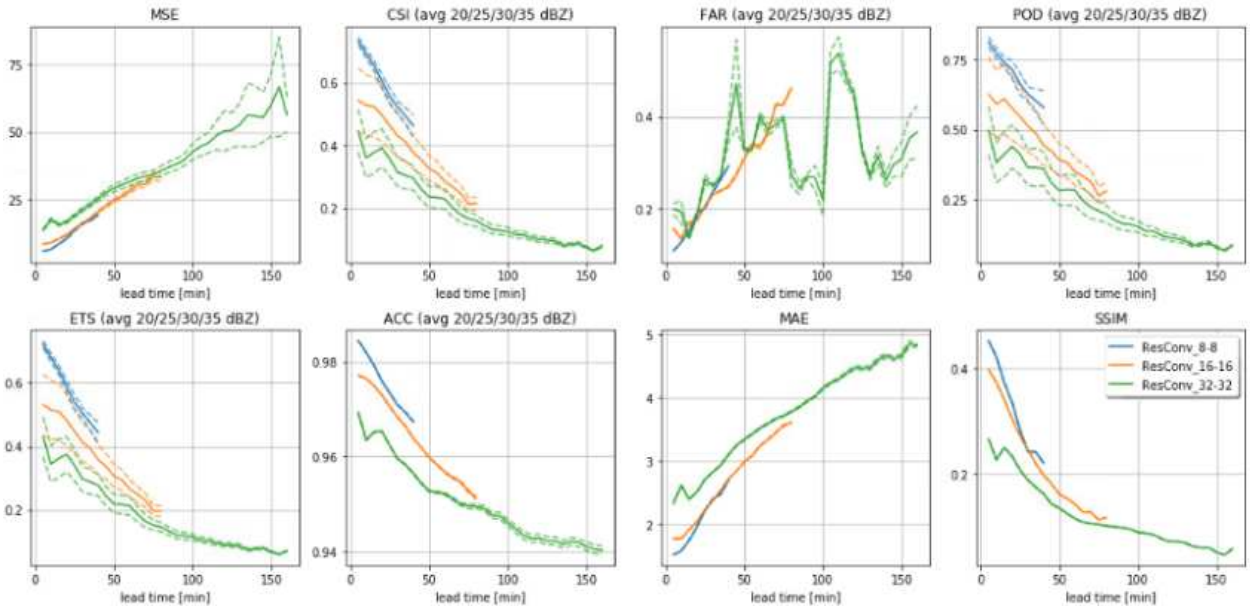


Figure 5.18: Metrics of different models varying the number of predicted frames.

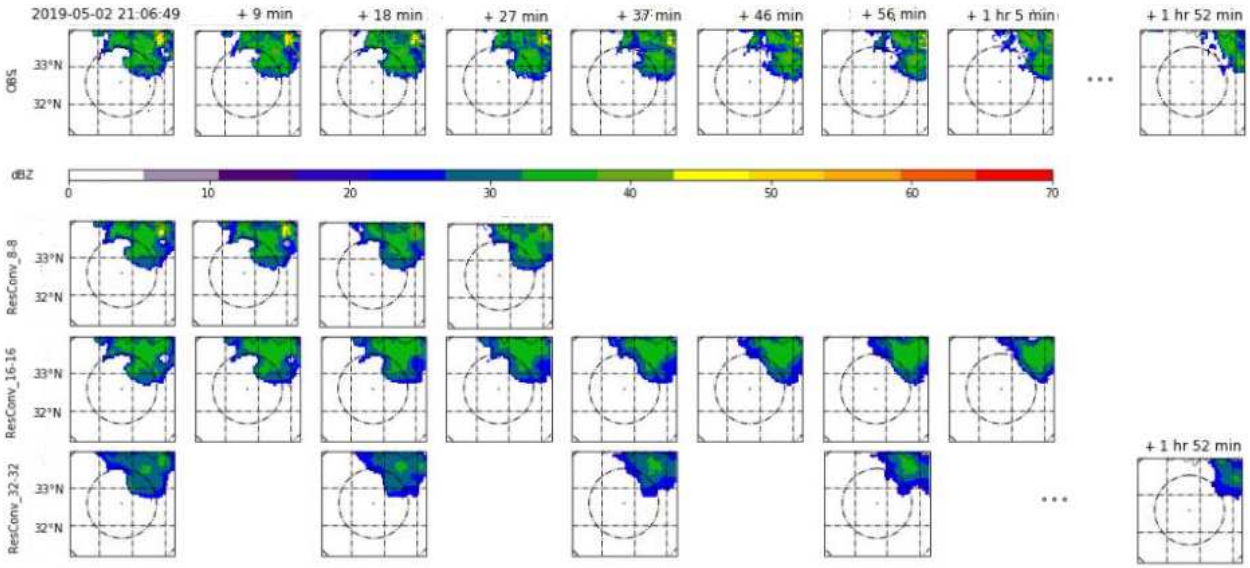


Figure 5.19: Prediction plots of models varying the number of predicted frames.

a model targeting shorter lead times if that is as far as one is interested in predicting, having a detriment in the predictions if longer lead times than desired are being forecast.

5.6 Behavior on different prediction steps

Like the previous experiment, here it is compared how the predictions changed when the sampling rate varies. For mimicking this, three models are used, ResConv_32-32, ResConv_16-16-2, and ResConv_16-16, where the number correspond to 'history length'-'prediction length'-('skip frames').

From the metrics in Figure 5.20 it can be observed the earlier frames are better predicted by the shorter-range predictor (ResConv-16-16), but it seems that it asymptotically matches the other two models. Between the two models covering the whole range of predictions, the one skipping every other frame seems to make slightly better predictions, but no definite winner can be drawn as both do better in some metrics, and the standard deviation ranges overlap at several scores.

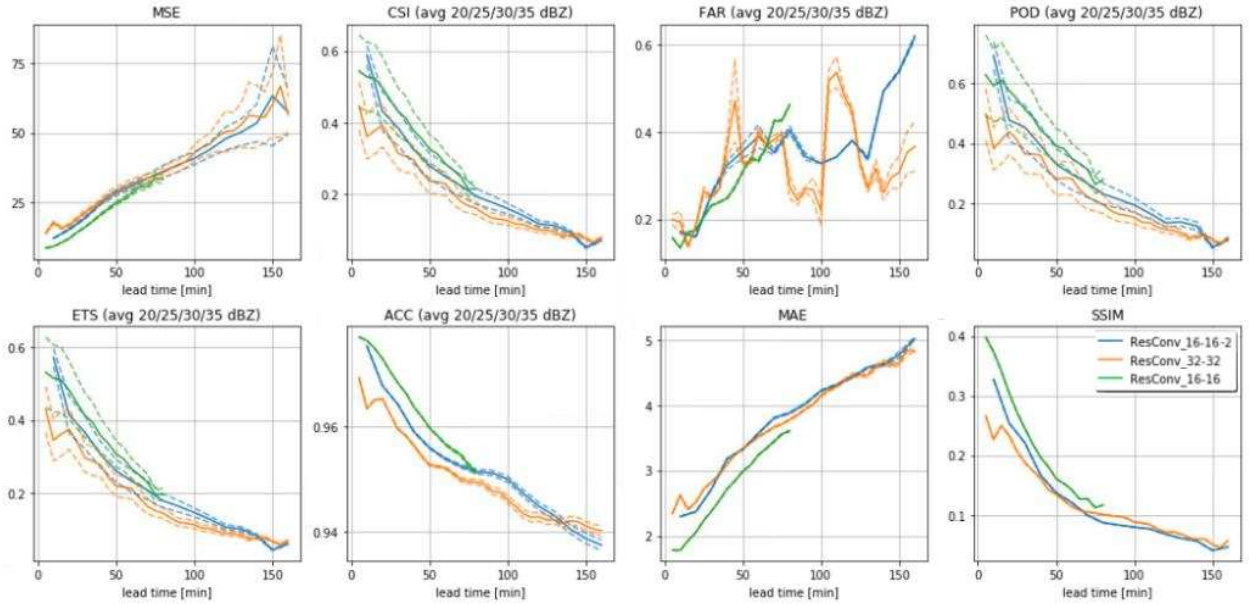


Figure 5.20: Metrics of models with different the step size .

The example in Figure 5.21 falls in line with the metrics above, where predicting fewer frames, by skipping or predicting less, seems to improve the predictions.

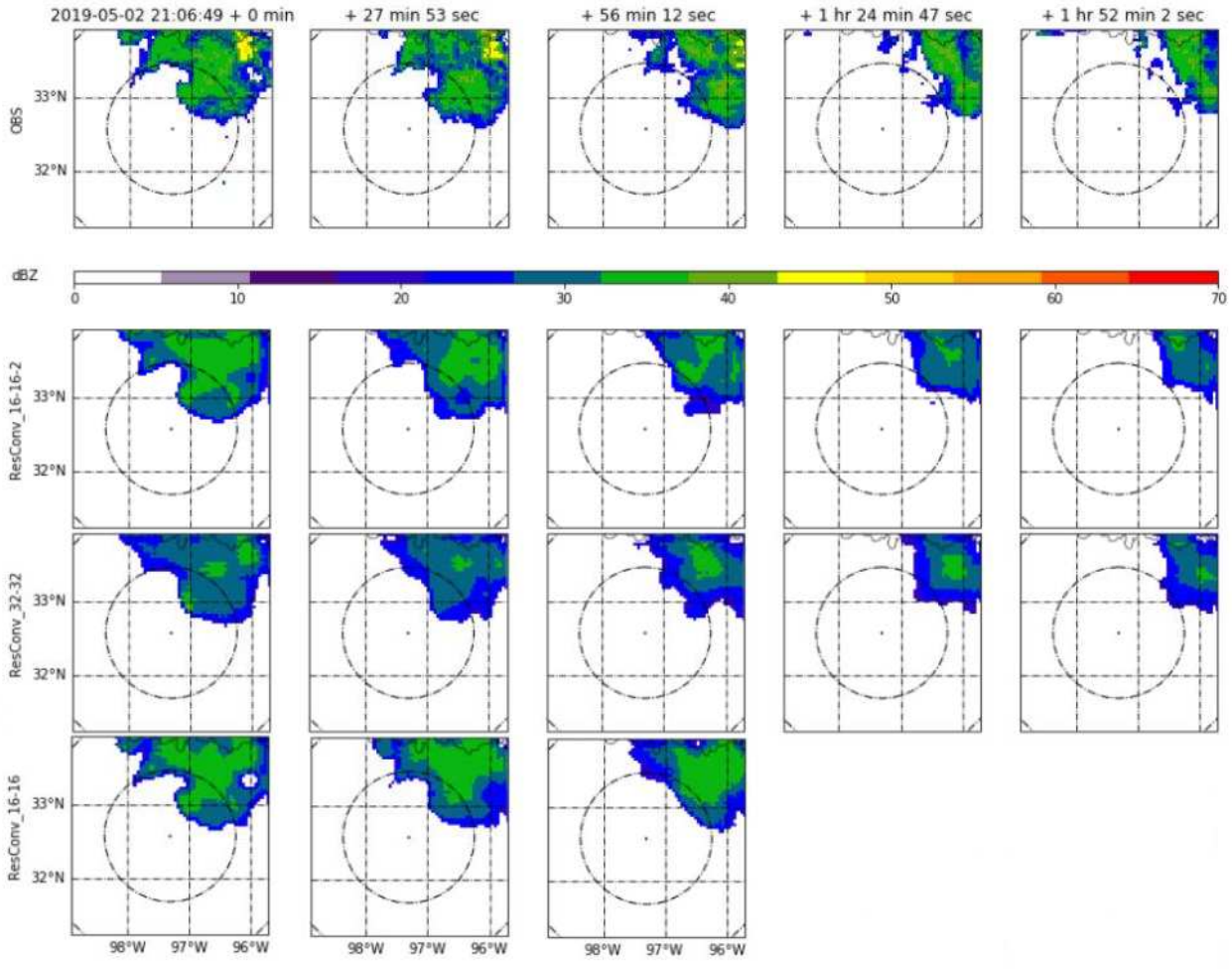


Figure 5.21: Prediction plots of models with different the step size .

5.7 Comparing proposed models against baseline models

The average metrics over different thresholds are shown in Figure 5.23. Here DARTS does better in the CSI and POD scores; ResConv, Composite, and ResGRU do better in MSE and FAR. RainNet does not perform well, and trajGRU does perform similar to the rest of the models at higher lead times but not at shorter lead times were it does worst. ResConv matches DARTS's mean CSI score at the last lead time, but the standard deviation ranges overlap at most of the lead times. DARTS has much smaller variation than ResConv, meaning that it performs similarly at every threshold while ResConv does not.

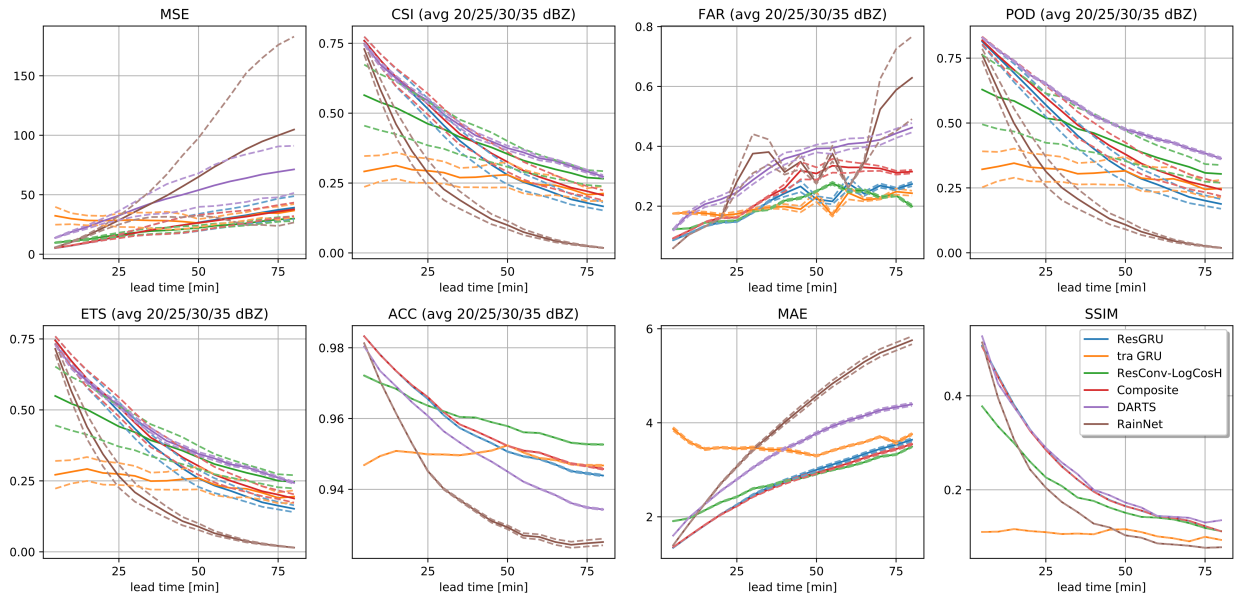


Figure 5.22: Comparison of average metrics over thresholds for baseline models and different architectures.

To analyze how they predict at different threshold values Figure 5.23 shows the CSI and FAR score for each. As expected, while ResConv does better at 20 dBZ and 25 dBZ, DARTS does considerably better at 35 dBZ. At 30 dBZ both do similar. ResGRU has a similar behavior as DARTS but with lower performance, and Composite (with ResGRU base) does just a bit better than ResGRU. By looking at FAR scores, DARTS does worst in every threshold.

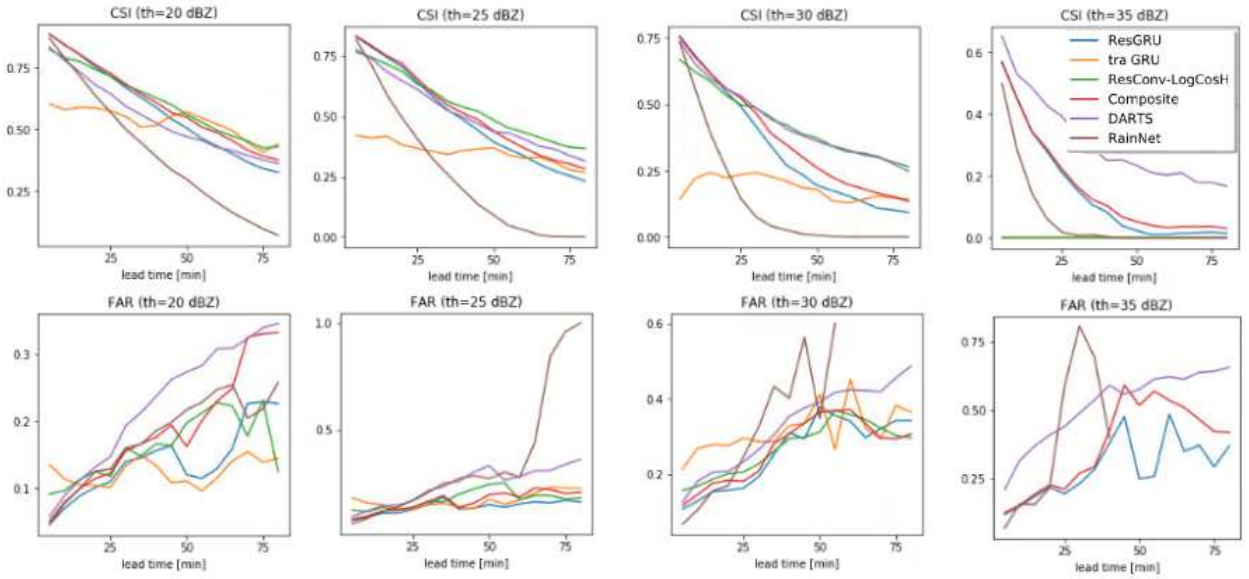


Figure 5.23: CSI and FAR scores at different thresholds for baseline models and proposed models.

Figure 5.24 show an example prediction for every model evaluated above. Visually, the results go with the metrics, being the most compelling predictions those done by DARTS, ResConv, and Composite. While RainNet seems to have good initial lead time predictions but deteriorates faster than the other models.

To further discuss the differences between DARTS and ResConv three more examples are presented. In Figure 5.25 a red area marks how ResConv successfully predicts the decay while DARTS seems to predict only the displacement of it. In Figure 5.26 once again, DARTS correctly predicts the advection but not the changes in shape as ResConv attempts to do if observed the last frame marked in red, or the sequence marked in magenta where DARTS maintains the 'Y' shape through the whole sequence. Neither model was able to predict the growth on the left part of the magenta section at the last two frames. Finally, in Figure 5.27 an example is presented where there is not too much translation but mainly growth. In this case, while DARTS mainly preserves shapes it does not predict any of the growth but ResConv does.

The following part of the experiment compares ResConv against all nowcasting implementation by PySteps which are all based on optical flow methods. Figure 5.28 shows the metrics, where Extrapolation has the best average score followed by ResConv, which does better in MSE and FAR.

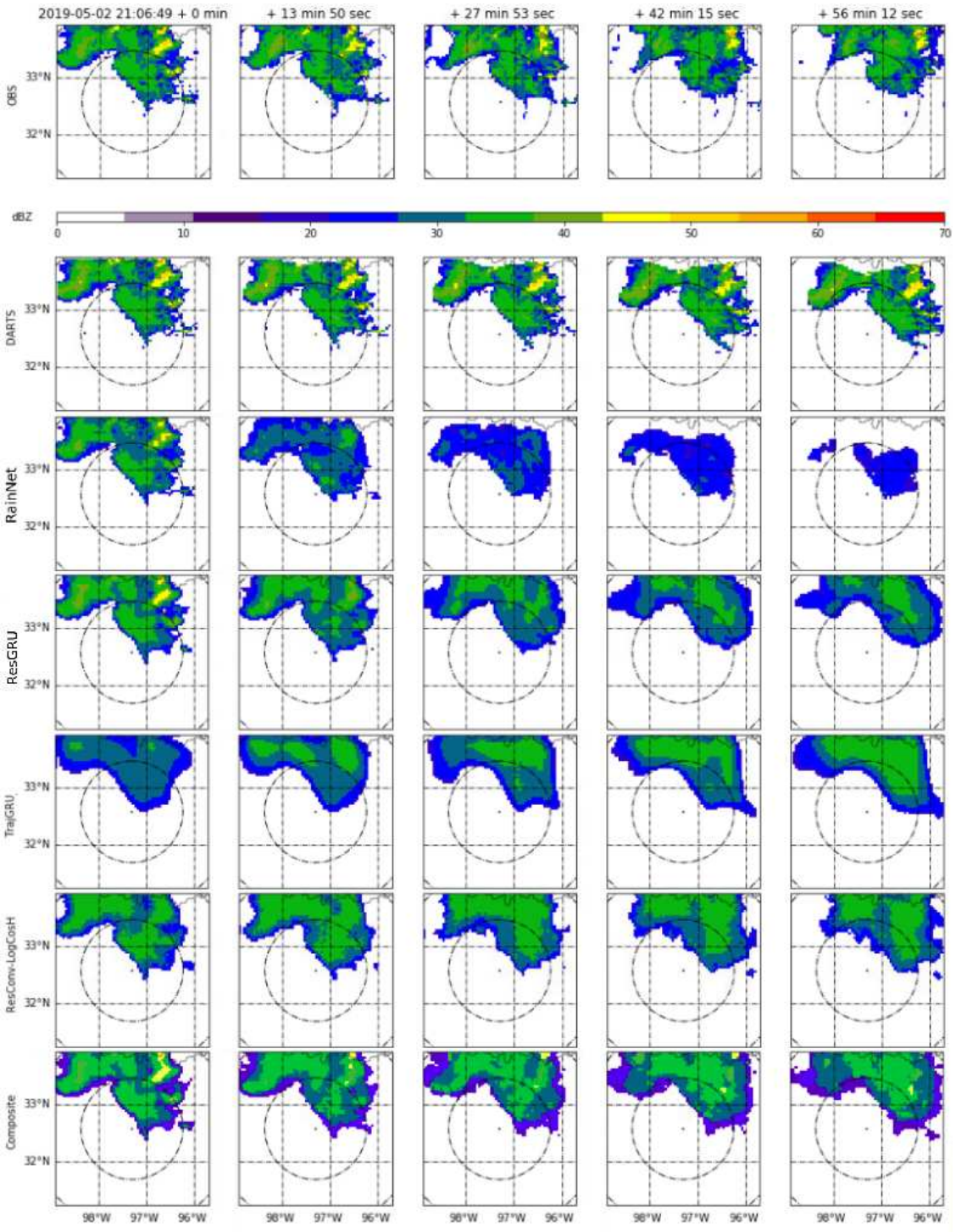


Figure 5.24: Prediction plots for baseline models and proposed models.

Like previous experiments, ResConv does better at lower reflectivity values but it performs poorly above 35 dBZ weakening the overall average. S-PROG and ANVIL did not perform well and it is

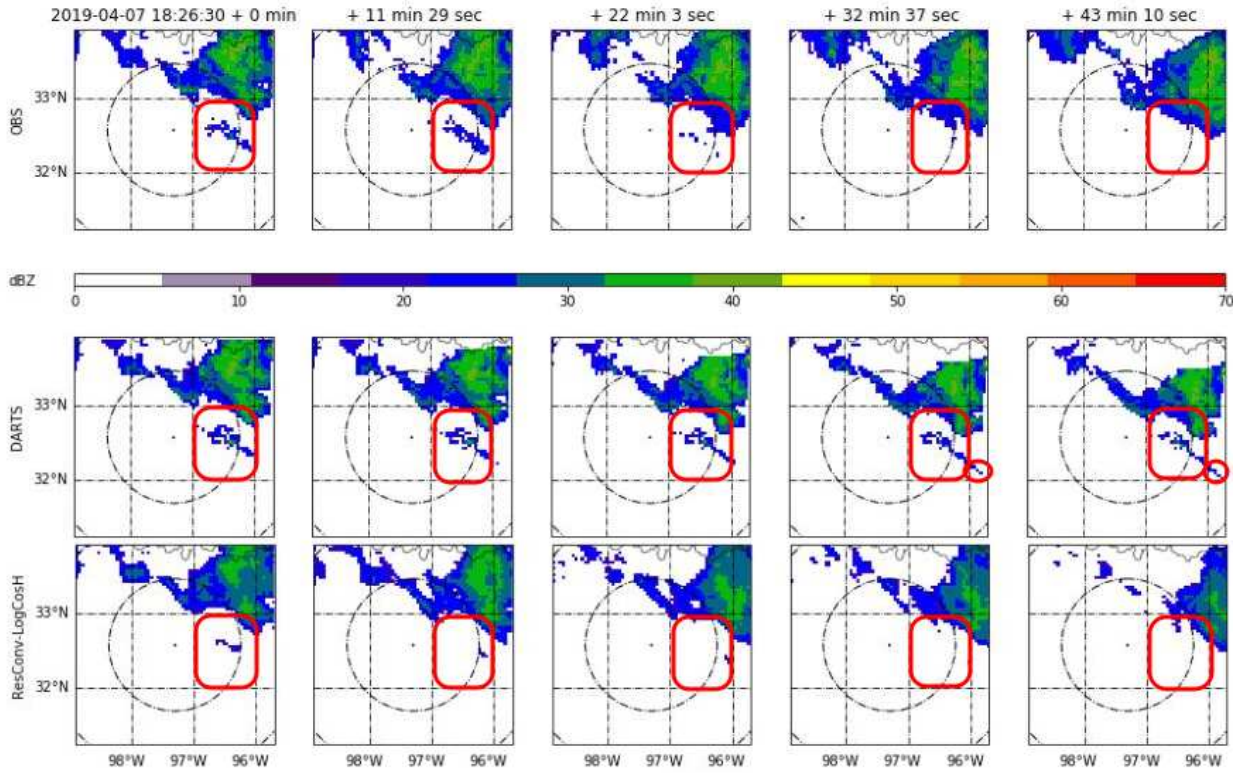


Figure 5.25: Prediction example 2 for DARTS and ResConv

more likely due to the incorrect usage of it rather than to the model itself, which has good results in the literature [75] and [18]. In Figure 5.29 a prediction example is presented.

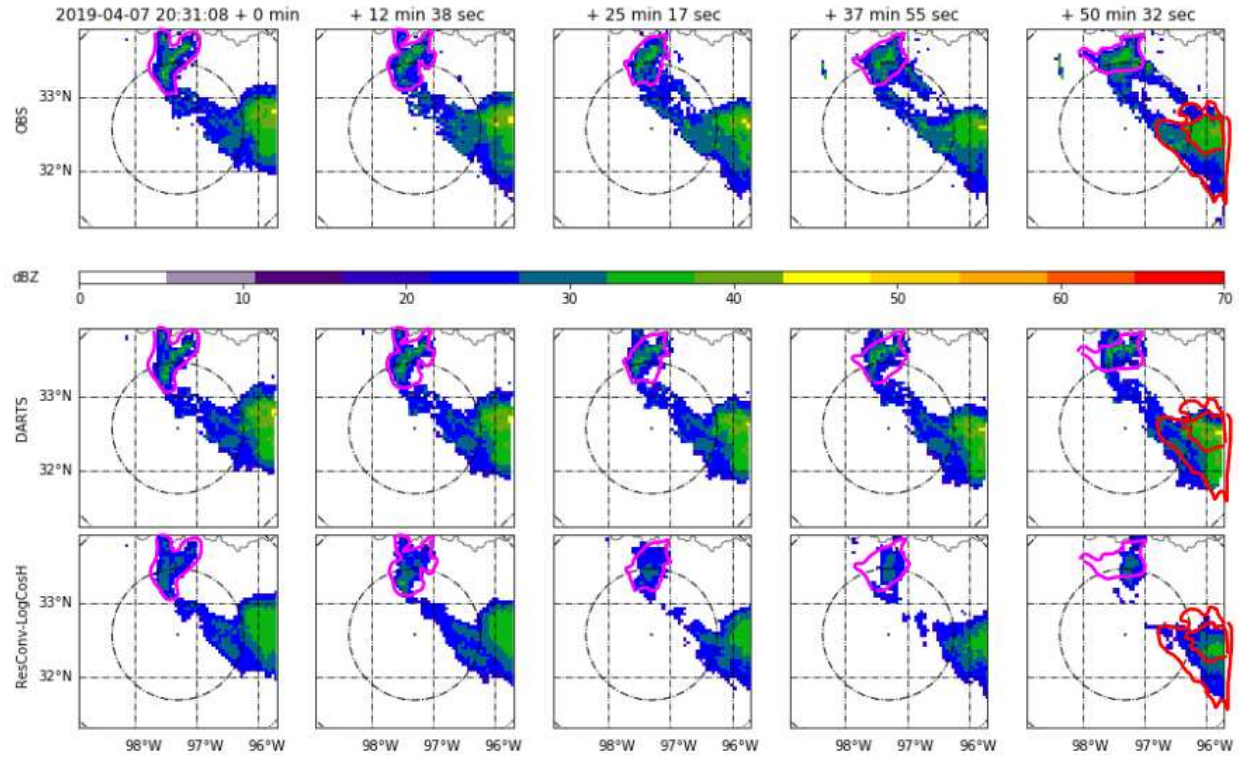


Figure 5.26: Prediction example 3 for DARTS and ResConv

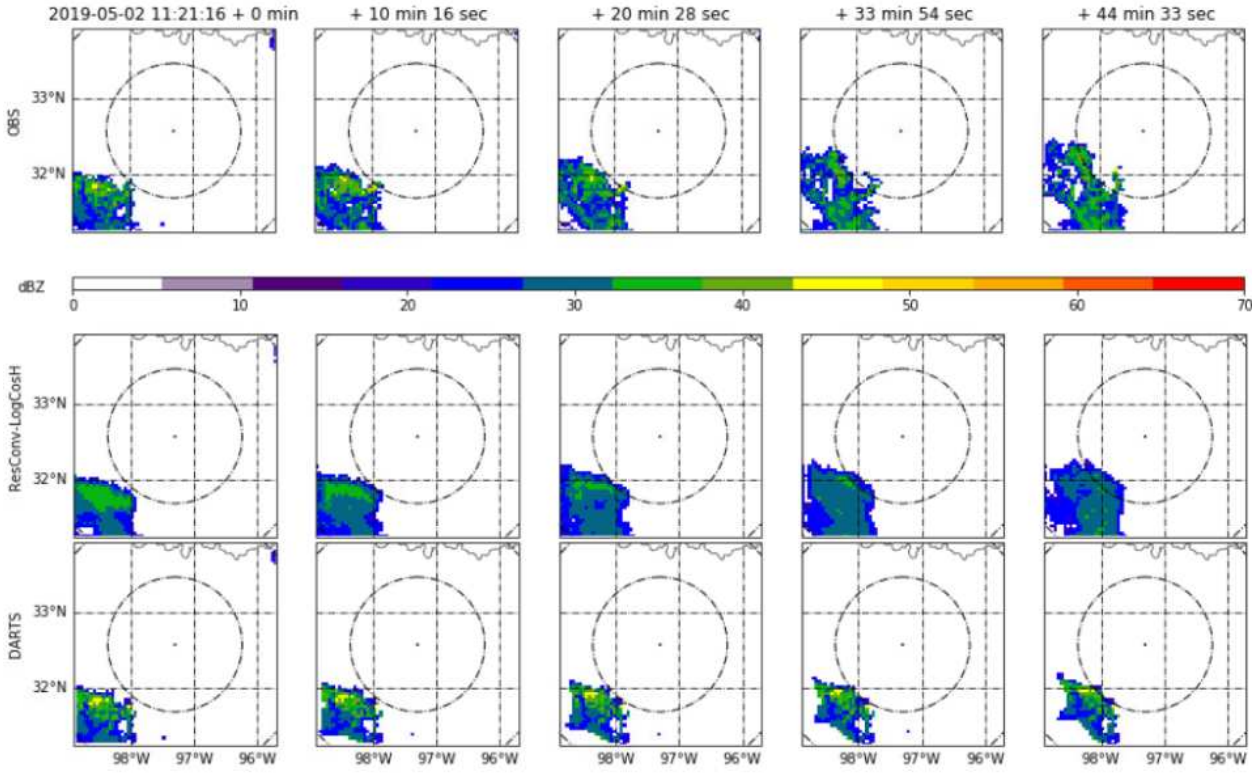


Figure 5.27: Prediction example 4 for DARTS and ResConv

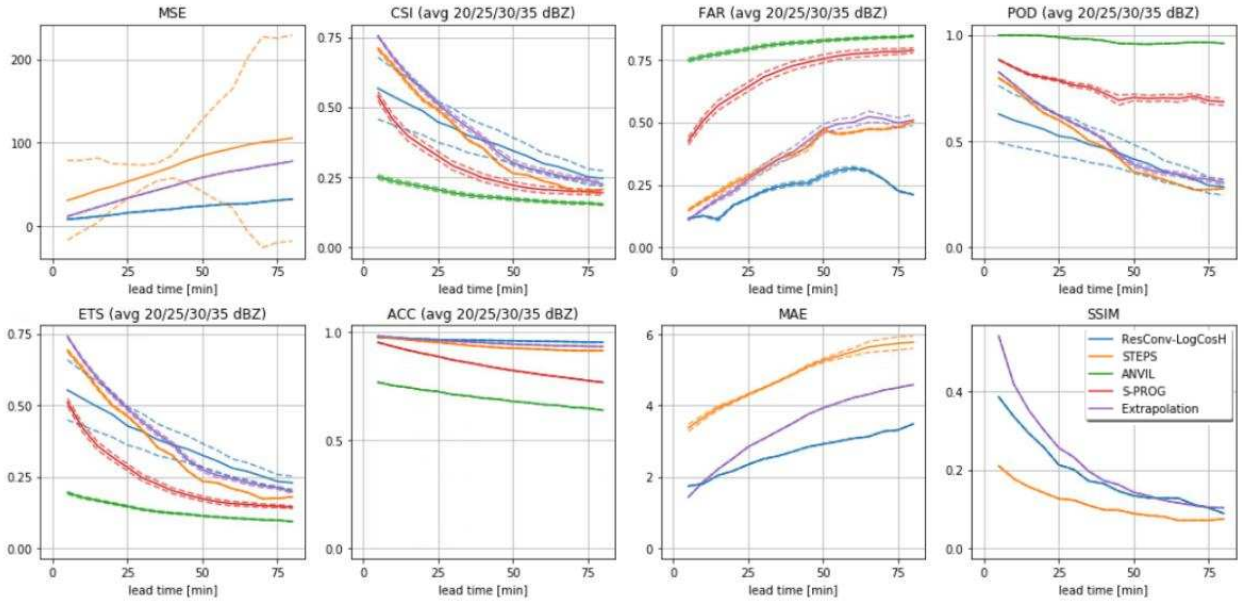


Figure 5.28: Comparison of average metrics over thresholds for PySteps models and ResConv.

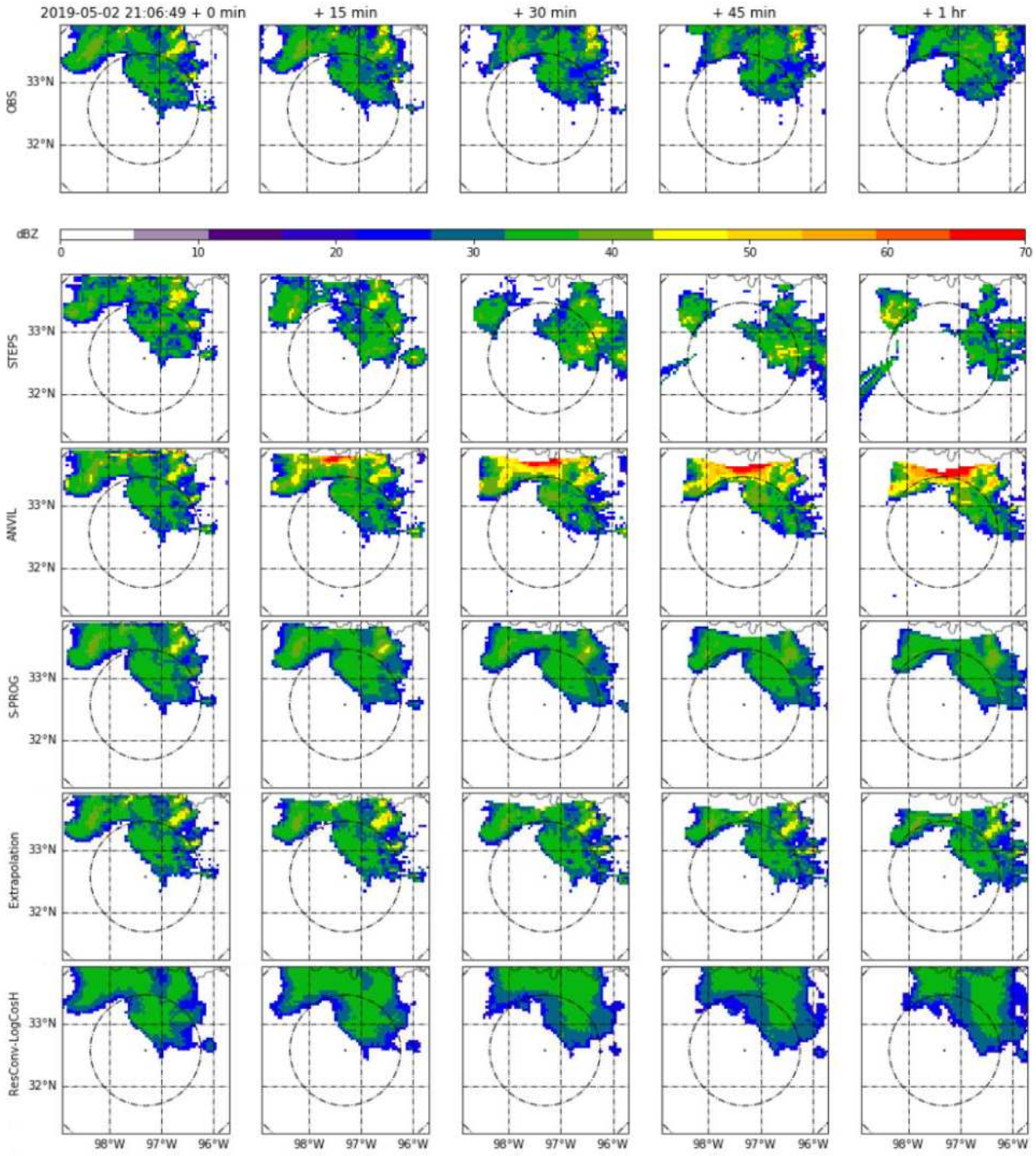


Figure 5.29: Prediction plots for PySteps models and ResConv.

5.8 Performance on different geographical regions

For these experiments, the metrics on both datasets and models, DARTS and ResConv, are presented in Figure 5.30. The relative performance between both models on each dataset seems to be consistent as both do worse in Denver predictions. To try to quantify this similarity the absolute difference between each dataset is computed and showed in Figure 5.31. The ranges of each metric are the same ranges as in Figure 5.30 or the metrics' theoretical ranges. The more each curve overlaps the more similar performance cross-datasets they have.

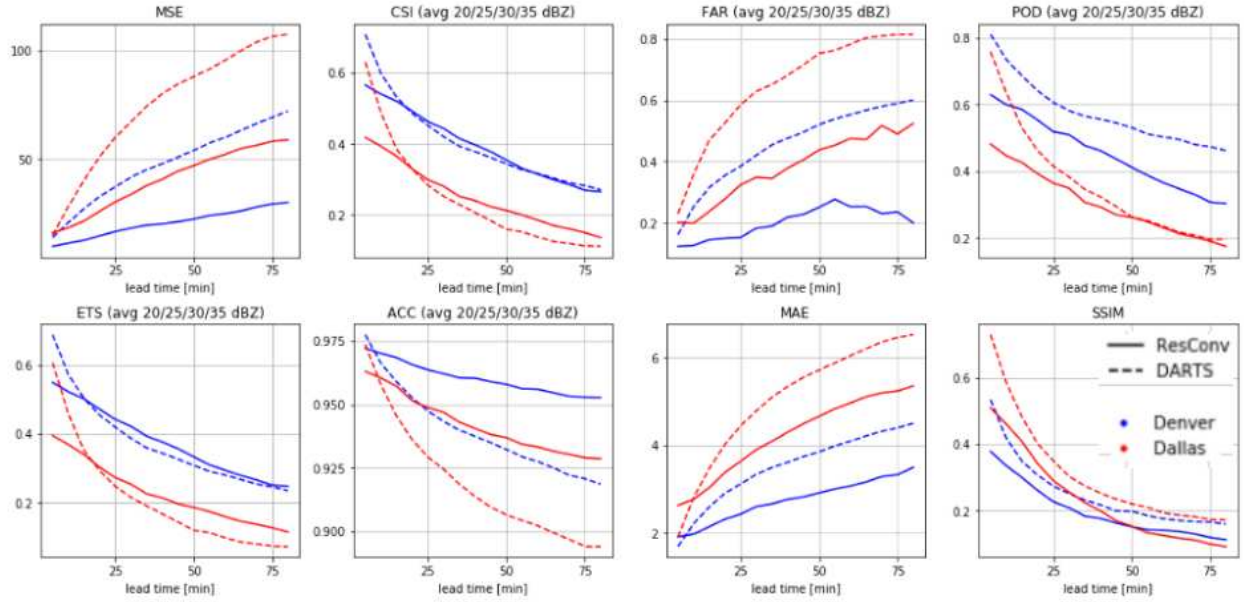


Figure 5.30: Comparison of average metrics over thresholds for different regions' datasets.

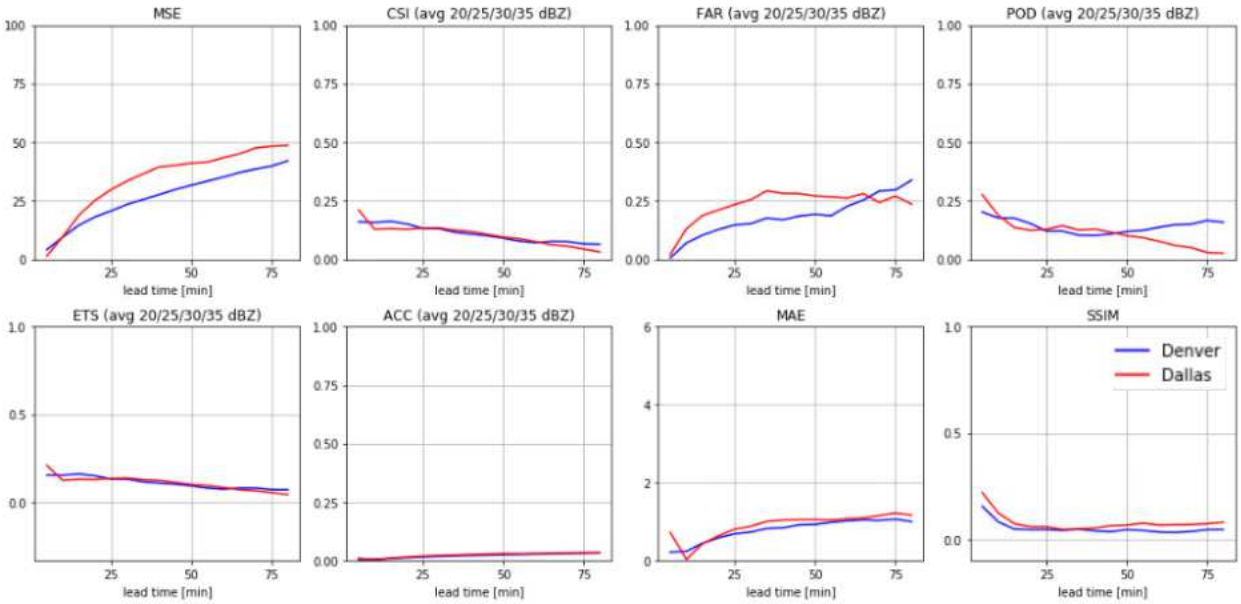


Figure 5.31: Comparison of average metrics over thresholds for different regions' datasets.

Chapter 6

Discussion

The aim of this research was to implement machine learning models for weather nowcasting purposes using radar echo data and compare them with preexisting models. Three different architectures were proposed to this end: a purely convolutional approach with residual connections, a mix of convolutional and convolution recurrent layers approach, and an ensemble of a continuous prediction with multiple discrete predictions. The process of creating these models consists of several stages where different analyses were done, many of which consist of trying to reproduce results from the literature, while others had not been done until now.

The major issue with using machine learning models, and particularly with the most common approach of training against a performance loss (another possible approach is using adversarial losses), is the blurriness effect. Consistently in every experiment, the predictions were not able to predict the highest values of the observations. This happens notably at longer lead times where there are more possible scenarios making the predictions more uncertain. A possible explanation is that the prediction could be interpreted as the average of all possible scenarios, thus getting blurrier as the standard deviation increases. Another factor that most likely contributes to this shortcome is the lack of high reflectivity values in the training dataset. It would make sense that the models are prone to predict values in the mid-range of the dataset.

Inherently connected with this is the performance loss used to train the machine learning models. Here arises the issue of not having a metric that measures all the aspects that can rank one prediction as 'better' than the other. This goes beyond weather nowcasting, as it is a limitation in many computer vision problems. Even if one metric was found, the convexity of it would be a constraint. For example, the binary metrics CSI and FAR, which are not convex cannot serve as a performance loss. That is why adversarial losses can be a promising approach as in a way the loss used to train a predictor is custom trained for that problem in spite of that in the background there is one traditional performance loss such as MSE. Nevertheless, for this reason, and also for the

fact that a different loss besides MSE could yield better predictions, different performance losses were tested. The results did not present an indisputable 'winner' as many of them did better on some aspects, but none in all. For example, B-MSE had an overall good performance and achieved better scores at longer lead times than MSE, but at the expense of overestimating the nowcast. The combination of metrics had excellent scores in the whole range of lead times but only at low reflectivity values. In Appendix A.1 the examples using different combinations had different outcomes showing that there might be a potential benefit if a right combination is found as in some cases the shape transformations are better predicted and in other the intensity values. LogCosH has an interesting behavior as it performs similarly at every reflectivity giving good scores both at CSI and FAR. The main limitation is that it was not able to reach the peak of 35 dBZ that MSE and B-MSE did, which affected the overall average. Nevertheless, based both on the metrics and the observed predictions, LogCosH was used in most of the experiments in the rest of the thesis. As many of the testing events had reflectivity values reaching close to 45 dBZ and almost none of the proposed model predicted values above 37 dBZ at the longest lead time, it seems fair to use a model not predicting anything at the last tested threshold (35 dBZ) when the comparisons should go up to 45 dBZ with the baseline models.

The only two models that presented high reflectivity values at long lead times were the Difference model, which predicts not the absolute values of each frame but the relative changes against the last history frame, and the Composite model, which consists of an ensemble of predictions at different thresholds. It is important to mention that the former was not optimized and a simple naive implementation was used. The apparent advantage is that it does not suffer from the blurriness effect, but it clearly does not predict correctly the shapes or even the advection. On the other hand, the Composite model, which to the best of the author's knowledge it has not been done before, outperforms the base model used. This positions the Composite as a great candidate, as knowing not only that given any model it can be self-improved by training it on different Z ranges, but also brings the possibility of using different models highly adapted to target those Z ranges.

Something that is uncommonly addressed in the literature, and requires a much deeper analysis that it was done here, is the study of how to optimize the predictions based on what to observe to make the predictions and how to make the predictions (how long and how spaced in time). It is common to follow the same conventions as previous authors used, such as the 10-10 (history-prediction frames) used by the original authors of convLSTM [19], which they probably inherited from the moving-MNIST dataset. In fact, originally the models used here were as such, but it seemed more natural to use a power of two for the upsampling and downsampling convolutions to avoid odd paddings. When using residual connections it is more convenient to use the same input lengths as output. Also, it could be reasonable to observed as far back as far ahead it is desired to predict, as both short-term and long-term dynamics are modeled. Nevertheless, it is more likely to depend on the data itself and it should be analyzed for every case. For more stochastic behaviors it might be logical just to observe a few frames in the past, while for a slow deterministic motion maybe longer context observations are better. The naive experiments done here suggest that using shorter history yields better predictions at short lead times, but longer history yields better predictions at longer lead times. The blurriness effect seems to increase not only by longer lead times but also by longer observed context. The experiments predicting up to different lead times strongly support that the overall predictions get affected by increasing the number of frames, meaning that it is recommended to predict up to the desired maximum lead time and not beyond. Finally, given the same period of time for both context and predictions, it seems to be better to decrease the sampling rate. This presents similar behavior as the above experiments, where less involved frames translate into sharper predictions. An assumption is that using the minimum required frames to satisfy the Nyquist equation is the best choice.

Different architectures were compared, where the results indicate that it is better to use purely convolutional layers without any recurrent layer when residual connections are used. An analogous result, not from the computer vision community, was presented here [76]. Besides yielding better scores, it was considerably faster to train a model where no RNN was used (GRU, LSTM, or trajGRU). On the comparison with baseline models, the experiments against DARTS are the

most important ones. DARTS does great predictions and outperforms the proposed methods in aspects like predicting high reflectivity values, and of course not suffering from the blurriness effect. Nevertheless, the FAR score, which indicates how much the predictions are being overestimated, is higher in DARTS than in ResConv. Also, DARTS is outperformed at lower reflectivity values. RainNet, a conceptually similar machine learning approach that was published four months prior to this writing, performed considerably worse but it is highly likely due to that is was designed for an input of 4 frames of 928x928 resolution, while here it was used with 16 frames of 64x64.

The last part of the experiments consisted of assessing the potential weakness of a machine learning approach, heavily dependent on the training dataset, which is using it to nowcast weather at a different geographical location. The results were promising in the sense that the performance did not deteriorate in a relative comparison with DARTS which is not geographical-dependent (at least not by definition).

Some of the advantages of using a machine learning model to make radar echo extrapolations are that it is fast to make the predictions, it does not depend on physics equations, it can harness the advances in the computer vision community, it could be trained specifically for a geographical location as well as a season of the year, and hopefully it could be used to do reverse engineering and provide insights on the physics behind hydrometer advection or other weather properties. On the downside, it is slow to train, it requires large history data and high computation power, and it is a black box, where no confidence values can be set for the predictions.

6.1 Future Work

There are multiple possible paths and tangents to further investigate on this topic. Machine learning is unquestionably a powerful technique that can improve current nowcasting systems, making the research on it worthy.

Machine learning models for other parts of a nowcasting system

Most of the current systems consist of ensembles of different models. With the comparisons done between ML and DARTS, it was seen that where one was deficient the other did well, meaning

that hopefully in combination they could improve the nowcasts. Machine learning can be used for both a standalone nowcasting system as it was done here, but it could also be used to estimate the optical flow and/or the advection in methods such as those in the PySteps project. For the Composite model, the ensemble of different predictions was done by taking the maximum value, but a ML model could also learn how to make the ensemble in a better way. For example, looking at the results obtained by ResConv and ResGRU, where at shorter lead times ResGRU was better, it could be interesting to see if an ensemble time-wise could be made for which at shorter lead times one model is used and at longer lead times another is used. This could easily be done by hand if the same model is used but trained at different lead times, but with different models the changes from one frame to the other will probably not be smooth.

Variation in the encoder-decoder architecture

This thesis was mostly based on the symmetrical encoder-decoder; other variations were presented but those were not optimized or analyzed. Besides the ones introduced here, other modifications can be teaching forcing, where the first part of the training consist of teaching how to encode the context and gradually transition to decode the predictions. Also, given the results obtained on shorter prediction sequences, it could be interesting to predict only one frame ahead and then recursively obtain the rest instead of predicting all at once.

In line with modifying the length of the context or prediction, an interesting study would be to apply techniques such as Deep Taylor Decomposition [77] or Layer-wise Relevance Propagation [78] to analyze how many frames are actually being used to make the predictions at different lead times.

Improving the dataset

As already mentioned, the training dataset is a key part of the system. Therefore, improving it would likely improve the predictions. Some ideas to build a better dataset are to increase the variability of it and to regularize the sequences. The former refers to having a dataset consisting mostly of the type of event for what a warning needs to be issued and containing events with growth

and decay; refer to Appendix A.2 for more details. Regularizing the dataset might bring benefits, too. It would be interesting to compare the performance of a model trained with a dataset with irregular sampling rate, with regular sampling rate across all events and regular sampling rate only within each event, but varying across events. Finally, another possibility is to do the predictions in polar coordinates (original coordinate system used by the radars), eliminating by this all artifacts that the gridding might cause.

Improving the model

The possibilities to improve the model are endless, but some of them could be regularization, activation functions, and convolutional filter sizes. For regularization, dropout layers were used as well as a weight decay factor, but the best results were with no regularization at all. Probably the reason is that the blurriness effect was covering all potential benefits of regularization. For the activation function, when RNNs were used LeakyReLu was key for them to work, and for the last activation function originally none was used but afterward with sigmoid function better results were obtained as some model predicts values beyond the target range and these were clipped out. Finally, the filter size of the convolutional filters was not a matter of optimization here and it could be potentially critical for obtaining sharper predictions.

Continuous learning

It is reasonable to think that for a weather nowcasting system, where the data changes seasonally, having a continuous-learning-deployed-system could see benefits compared to a static one.

6.2 Conclusion

In this thesis, the bases for future researches in weather nowcasting were laid. Firstly, a thorough literature review was done going beyond just nowcasting-related works, but also addressing the more general video prediction problem in computer vision. Then, three different models were proposed and compared to several baseline models achieving the state of the art results. Some of the literature proposals, such as different losses, were tested as well as new experiments, such as the

performance on different lead times and geographical regions. Finally, and most importantly, the work done here can be almost entirely reproduced as all the code is open-source and available [66], except for DARTS predictions which were done externally.

Bibliography

[1] Steven M. Hunter. WSR-88D Radar Rainfall Estimation: Capabilities, Limitations and Potential Improvements. *National Weather Service Office*, 20:26–36, 1996.

[2] Shawn Milrad. Radar Imagery. *Synoptic Analysis and Forecasting*, pages 163–177, 2018.

[3] Richard J. Doviak and Dušan S. Zrnić. Precipitation Measurements. In *Doppler Radar and Weather Observations*, pages 209–279. Elsevier, Jan 1993.

[4] WMO. *Guidelines for Nowcasting Techniques*. Number 1198. 2017.

[5] Michael Dixon and Gerry Wiener. TITAN: Thunderstorm Identification, Tracking, Analysis, and Nowcasting—A Radar-based Methodology. *Journal of Atmospheric and Oceanic Technology*, 10(6):785–797, 12 1993.

[6] J. T. Johnson, Pamela L. MacKeen, Arthur Witt, E. De Wayne Mitchell, Gregory J. Stumpf, Michael D. Eilts, and Kevin W. Thomas. The Storm Cell Identification and Tracking Algorithm: An Enhanced WSR-88D Algorithm. *Weather and Forecasting*, 13(2):263–276, 06 1998.

[7] Alessandro M. Hering, Christian Morel, Gianmario Galli, Paolo Ambrosetti, and Marco Boscacci. Nowcasting thunderstorms in the alpine region using a radar based adaptive thresholding scheme. 2004.

[8] Sung-Hwa Jung and Gyuwon Lee. Radar-based cell tracking with fuzzy logic approach. *Meteorological Applications*, 22, 07 2015.

[9] L. Li, Willi Schmid, and J. Joss. Nowcasting of motion and growth of precipitation with radar over a complex orography. *Journal of Applied Meteorology - J APPL METEOROL*, 34:1286–1300, 06 1995.

[10] Urs Germann and Isztar Zawadzki. Scale-Dependence of the Predictability of Precipitation from Continental Radar Images. Part I: Description of the Methodology. *Monthly Weather Review*, 130(12):2859–2873, 12 2002.

[11] Neill Bowler, Clive Pierce, and Alan Seed. Development of a rainfall nowcasting algorithm based on optical flow techniques. *Journal of Hydrology*, 288:74–91, 03 2004.

[12] Evan Ruzanski, V. Chandrasekar, and Yanting Wang. The casa nowcasting system. *Journal of Atmospheric and Oceanic Technology - J ATMOS OCEAN TECHNOL*, 28:640–655, 05 2011.

[13] C. Mueller, T. Saxen, R. Roberts, J. Wilson, T. Betancourt, S. Dettling, N. Oien, and J. Yee. NCAR Auto-Nowcast System. *Weather and Forecasting*, 18(4):545–561, 08 2003.

[14] P.W Li and Edwin Lai. Short-range quantitative precipitation forecasting in hong kong. *Journal of Hydrology*, 288:189–209, 03 2004.

[15] Paul M. James, Bernhard K. Reichert, and Dirk Heizenreder. NowCastMIX: Automatic Integrated Warnings for Severe Convection on Nowcasting Time Scales at the German Weather Service. *Weather and Forecasting*, 33(5):1413–1433, 10 2018.

[16] George Alexander Isaac, Monika Bailey, Faisal S. Boudala, William R. Burrows, Stewart Cober, Robert Crawford, Norman Donaldson, Ismail Gultepe, Bjarne Hansen, Ivan Heckman, Laura X. Huang, Alister Ling, Jocelyn Mailhot, Jason A. Milbrandt, Janti Reid, and Marc Fournier. The canadian airport nowcasting system (can-now). *Meteorological Applications*, 21:30–49, 2014.

[17] T. Haiden, A. Kann, C. Wittmann, G. Pistotnik, B. Bica, and C. Gruber. The Integrated Nowcasting through Comprehensive Analysis (INCA) System and Its Validation over the Eastern Alpine Region. *Weather and Forecasting*, 26(2):166–183, 04 2011.

[18] Neill E. Bowler, Clive E. Pierce, and Alan W. Seed. STEPS: A probabilistic precipitation forecasting scheme which merges an extrapolation nowcast with downscaled NWP. *Quarterly Journal of the Royal Meteorological Society*, 132(620):2127–2155, oct 2006.

[19] Xingjian Shi, Zhourong Chen, Hao Wang, Dit Yan Yeung, Wai Kin Wong, and Wang Chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Advances in Neural Information Processing Systems*, pages 802–810, 2015.

[20] En Shi, Qian Li, Daquan Gu, and Zhangming Zhao. A Method of Weather Radar Echo Extrapolation Based on Convolutional Neural Networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10704 LNCS, pages 16–28. Springer Verlag, 2018.

[21] Ata Akbari Asanjan, Tiantian Yang, Kuolin Hsu, Soroosh Sorooshian, Junqiang Lin, and Qidong Peng. Short-Term Precipitation Forecast Based on the PERSIANN System and LSTM Recurrent Neural Networks. *Journal of Geophysical Research: Atmospheres*, 123(22):12,543–12,563, 2018.

[22] Xiaodan Liang, Lisa Lee, Wei Dai, and Eric P. Xing. Dual Motion GAN for Future-Flow Embedded Video Prediction. Technical report, Institute of Electrical and Electronics Engineers Inc., 2017.

[23] Jinrui Jing, Qian Li, and Xuan Peng. MLC-LSTM: Exploiting the spatiotemporal correlation between multi-level weather radar echoes for echo sequence extrapolation. *Sensors (Switzerland)*, 19(18), sep 2019.

[24] Quang Khai Tran and Sa Kwang Song. Computer vision in precipitation nowcasting: Applying image quality assessment metrics for training deep neural networks. *Atmosphere*, 10(5):1–20, 2019.

[25] Vladimíra Hežeľová. Artificial Neural Network for Precipitation Nowcasting. Master's thesis, Masaryk University, Faculty of Informatics, 2018.

[26] Paul R. Goldin. Review paper. *Journal of Chinese Philosophy*, 38(2):328–329, 2011.

[27] Quang-Khai ; Song Tran and Sa-kwang. Multi-ChannelWeather Radar Echo Extrapolation
 with Convolutional Recurrent Neural Networks. *Remote Sensing*, 11(19):2303, oct 2019.

[28] Chen, Zhang, Liu, and Zeng. Generative Adversarial Networks Capabilities for Super-Resolution Reconstruction of Weather Radar Echo Images. *Atmosphere*, 10(9):555, sep 2019.

[29] Xingjian Shi, Zhihan Gao, Leonard Lausen, Hao Wang, Dit Yan Yeung, Wai Kin Wong, and Wang Chun Woo. Deep learning for precipitation nowcasting: A benchmark and a new model. *Advances in Neural Information Processing Systems*, pages 5618–5628, 2017.

[30] Wang-chun Woo and Wai Kin Wong. Operational application of optical flow techniques to radar-based rainfall nowcasting. *Atmosphere*, 2017:48, 02 2017.

[31] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised Learning of Video Representations using LSTMs. *arXiv e-prints*, page arXiv:1502.04681, February 2015.

[32] Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc Van Gool. Dynamic Filter Networks. *arXiv e-prints*, page arXiv:1605.09673, May 2016.

[33] William Lotter, Gabriel Kreiman, and David Cox. Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning. *arXiv e-prints*, page arXiv:1605.08104, May 2016.

[34] Ryoma Sato, Hisashi Kashima, and Takehiro Yamamoto. Short-term precipitation prediction with skip-connected PredNet. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11141 LNCS, pages 373–382. Springer Verlag, oct 2018.

[35] Jinrui Jing, Qian Li, Xinya Ding, Nengli Sun, Rong Tang, Yali Cai, Radar Echo Extrapolation, Deep Learning, Recurrent Neural Network, Generative Adversarial, Adversarial Train-

ing, and Short-term Weather Forecasting. AENN : a generative adversarial neural network for weather. XLII:25–27, 2019.

[36] Georgy Ayzel, Maik Heistermann, and Tanja Winterrath. Optical flow models as an open benchmark for radar-based precipitation nowcasting (rainymotion v0.1). *Geoscientific Model Development Discussions*, pages 1–23, 09 2018.

[37] Shreya Agrawal, Luke Barrington, Carla Bromberg, John Burge, Cenk Gazen, and Jason Hickey. Machine Learning for Precipitation Nowcasting from Radar Images. *arXiv e-prints*, page arXiv:1912.12132, December 2019.

[38] Georgy Ayzel, Tobias Scheffer, and Maik Heistermann. RainNet v1.0: a convolutional neural network for radar-based precipitation nowcasting. *Geoscientific Model Development*, (March):1–20, 2020.

[39] Lei Han, Juanzhen Sun, Wei Zhang, Yuanyuan Xiu, Hailei Feng, and Yinjing Lin. A Machine Learning Nowcasting Method based on Real-time Reanalysis Data. *Journal of Geophysical Research*, 122(7):4038–4051, sep 2016.

[40] G. Ayzel, M. Heistermann, A. Sorokin, O. Nikitin, and O. Lukyanova. All convolutional neural networks for radar-based precipitation nowcasting. In *Procedia Computer Science*, volume 150, pages 186–192. Elsevier B.V., jan 2019.

[41]

[42] Aifang Su, Han Li, Liman Cui, and Yungang Chen. A convection nowcasting method based on machine learning. *Advances in Meteorology*, 2020:1–13, 01 2020.

[43] Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee. Learning to Generate Long-term Future via Hierarchical Prediction. *arXiv e-prints*, page arXiv:1704.05831, April 2017.

[44] Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S. Yu. Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. In *NIPS*, 2017.

[45] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. GP-GAN: Towards Realistic High-Resolution Image Blending. *arXiv e-prints*, page arXiv:1703.07195, March 2017.

[46] D. Pavlyuk. Spatiotemporal traffic forecasting as a video prediction problem. In *2019 6th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 1–7, 2019.

[47] Junyan Wang, Bingzhang Hu, Yang Long, and Yu Guan. Order Matters: Shuffling Sequence Generation for Video Prediction. *arXiv e-prints*, page arXiv:1907.08845, July 2019.

[48] Matin Hosseini, Anthony S. Maida, Majid Hosseini, and Gottumukkala Raju. Inception-inspired LSTM for Next-frame Video Prediction. *arXiv e-prints*, page arXiv:1909.05622, August 2019.

[49] Alex X. Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic Adversarial Video Prediction. *arXiv e-prints*, page arXiv:1804.01523, April 2018.

[50] Marc Oliu, Javier Selva, and Sergio Escalera. Folded Recurrent Neural Networks for Future Video Prediction. *arXiv e-prints*, page arXiv:1712.00311, December 2017.

[51] Wen Liu, Weixin Luo, Dongze Lian, and Shenghua Gao. Future Frame Prediction for Anomaly Detection – A New Baseline. *arXiv e-prints*, page arXiv:1712.09867, December 2017.

[52] Henglai Wei, Xiaochuan Yin, and Penghong Lin. Novel Video Prediction for Large-scale Scene using Optical Flow. *arXiv e-prints*, page arXiv:1805.12243, May 2018.

[53] Niloofar Azizi, Hafez Farazi, and Sven Behnke. Location Dependency in Video Prediction. *arXiv e-prints*, page arXiv:1810.04937, October 2018.

[54] Viorica Patraucean, Ankur Handa, and Roberto Cipolla. Spatio-temporal video autoencoder with differentiable memory. *arXiv e-prints*, page arXiv:1511.06309, November 2015.

[55] Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Learning to Decompose and Disentangle Representations for Video Prediction. *arXiv e-prints*, page arXiv:1806.04166, June 2018.

[56] Sandra Aigner and Marco Körner. FutureGAN : Anticipating the Future Frames of Video Sequences using Spatio-Temporal 3d Convolutions in Progressively Growing GANs arXiv : 1810 . 01325v2 [cs . CV] 26 Nov 2018. 2018.

[57] Ziru Xu, Yunbo Wang, Mingsheng Long, and Jianmin Wang. Predcnn: Predictive learning with cascade convolutions. pages 2940–2947, 07 2018.

[58] Hafez Farazi and Sven Behnke. Frequency Domain Transformer Networks for Video Prediction. *arXiv e-prints*, page arXiv:1903.00271, March 2019.

[59] Nelly Elsayed, Anthony S. Maida, and Magdy Bayoumi. Reduced-Gate Convolutional LSTM Using Predictive Coding for Spatiotemporal Prediction. *arXiv e-prints*, page arXiv:1810.07251, October 2018.

[60] Yunbo Wang, Lu Jiang, Ming-Hsuan Yang, Li-Jia Li, Mingsheng Long, and Li Fei-Fei. Eidetic 3d lstm: A model for video prediction and beyond. In *ICLR*, 2019.

[61] Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. VideoFlow: A Conditional Flow-Based Model for Stochastic Video Generation. *arXiv e-prints*, page arXiv:1903.01434, March 2019.

[62] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[63] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations us-

ing RNN Encoder-Decoder for Statistical Machine Translation. *arXiv e-prints*, page arXiv:1406.1078, June 2014.

[64] Jonathan J. Helmus and Scott Collis. The python arm radar toolkit (py-art), a library for working with weather radar data in the python programming language. *Journal of open research software*, 4, 2016.

[65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, pages 770–778. IEEE Computer Society, dec 2016.

[66] Joaquin Cuomo. ML Nowcasting repository, 2020. https://github.com/JCuomo/NowCasting.

[67] Dominique Brunet, Edward R. Vrscay, and Zhou Wang. On the mathematical properties of the structural similarity index. *IEEE Transactions on Image Processing*, 2012.

[68] Barbara Brown, Randy Bullock, Tressa Fowler, John Halley Gotway, Kathryn Newman, Tara Jensen, Karin Meier-fleischer, David C. Carslaw, and Karl Ropkins. Developmental Testbed Center Boulder, Colorado. *Environmental Modelling and Software*, 27-28(April):52–61, 2014.

[69] Seppo Pulkkinen, Daniele Nerini, Andrés A. Pérez Hortal, Carlos Velasco-Forero, Alan Seed, Urs Germann, and Loris Foresti. Pysteps: an open-source python library for probabilistic precipitation nowcasting (v1.0). 2019.

[70] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9351, pages 234–241. Springer Verlag, may 2015.

[71] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, dec 2017.

[72] Xingjian Shi. Code repository from [19], 2017. https://github.com/sxjscience/HKO-7.

[73] Zhizhong Huang. trajGRU implementation in PyTorch. 2019.

[74] J. Bergstra, D. Yamins, and D. D. Cox. Making a Science of Model Search. *arXiv e-prints*, page arXiv:1209.5111, September 2012.

[75] S. Pulkkinen, V. Chandrasekar, A. von Lerber, and A. Harri. Nowcasting of convective rainfall using volumetric radar observations. *IEEE Transactions on Geoscience and Remote Sensing*, pages 1–15, 2020.

[76] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv e-prints*, page arXiv:1803.01271, March 2018.

[77] Grégoire Montavon, Tu-berlinde Sebastian Bach, Hhifraunhoferde Alexander Binder, Alexander Binder, Wojciech Samek, and Hhifraunhoferde M Klaus-Robert. Deep Taylor Decomposition of Neural Networks. Technical report.

[78] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus Robert Müller. Layer-Wise Relevance Propagation: An Overview. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11700 LNCS, pages 193–209. Springer Verlag, 2019.

[79] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and Checkerboard Artifacts, 2016.

[80] J. L. Pech-Pacheco, G. Cristöbal, J. Chamorro-Martínez, and J. Fernândez-Valdivia. Diatom autofocusing in brightfield microscopy: A comparative study. *Proceedings - International Conference on Pattern Recognition*, 15(3):314–317, 2000.

[81] Ivan and Laptev and Barbara Caputo. KTH Human Actions dataset. https://www.csc.kth.se/cvap/actions/.

Appendix A

Appendix

A.1 Losses

Three different Combined models are tested here. Each of them have the following loss function:

- Combined 1: $Loss = 1 \cdot MSE + 0.1 \cdot MAE + 0.02 \cdot (1 - SSIM)$
- Combined 2: $Loss = 1 \cdot LogCosH + 0.1 \cdot MAE + 0.02 \cdot (1 - SSIM)$
- Combined 3: $Loss = 1 \cdot LogCosH + 0.5 \cdot MAE + 0.03 \cdot MSE$

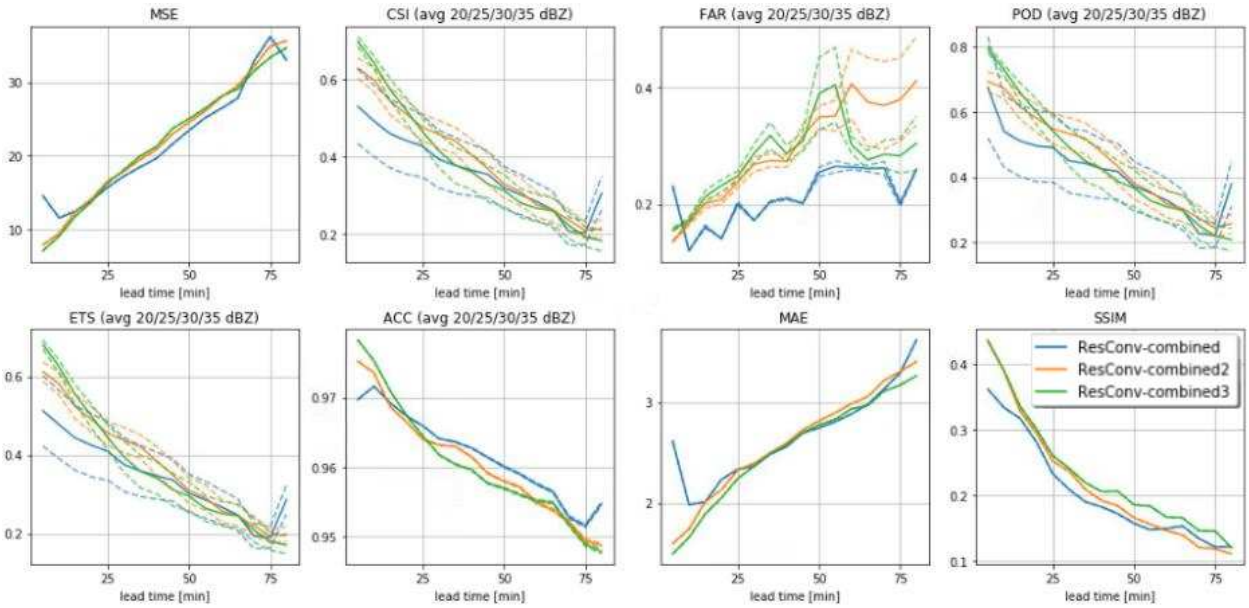


Figure A.1: Metrics using three different combined losses.

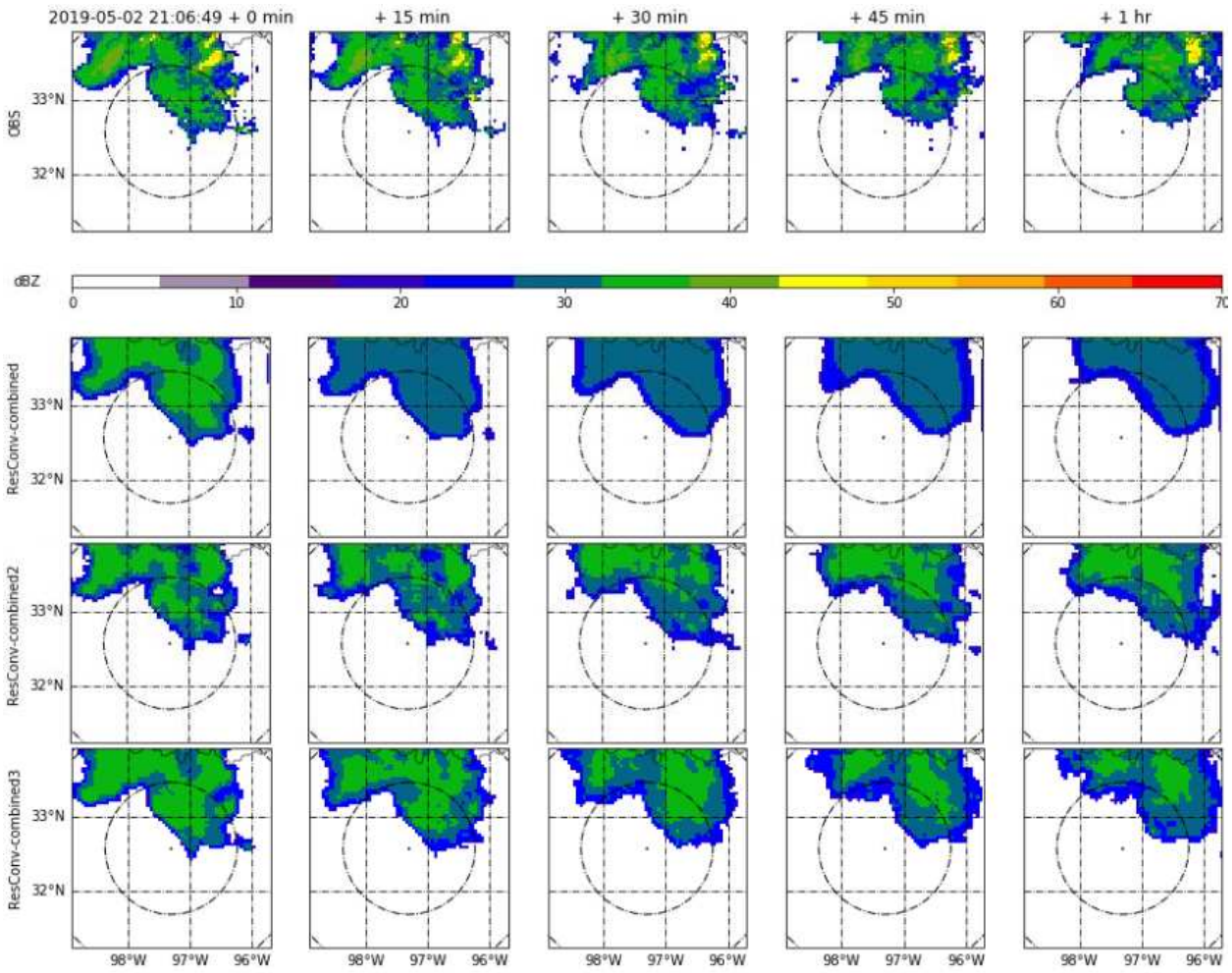


Figure A.2: Example plots using three different combined losses.

A.2 Training datasets

To support the idea that improving the dataset would probably improve the nowcast in Figure A.3 the histogram of reflectivity content per frame, including the history and the observation, is shown. As well, in Figure A.4 an event example is shown with its corresponding histogram to contrast how ideally the histogram overall the dataset should look like.

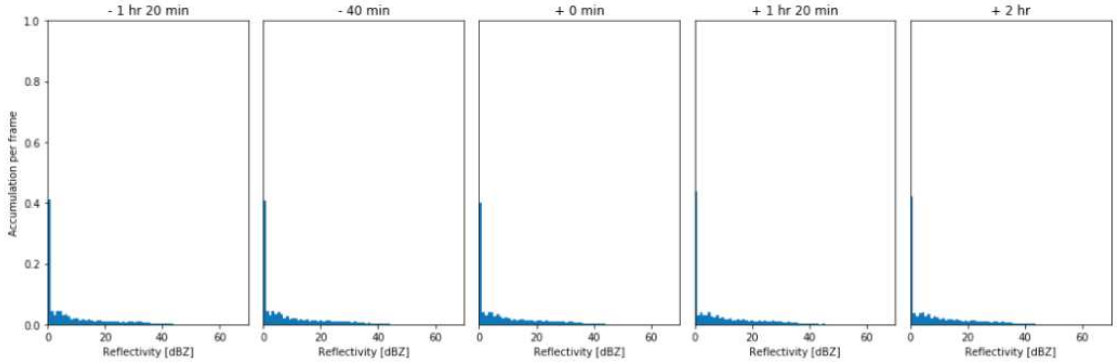
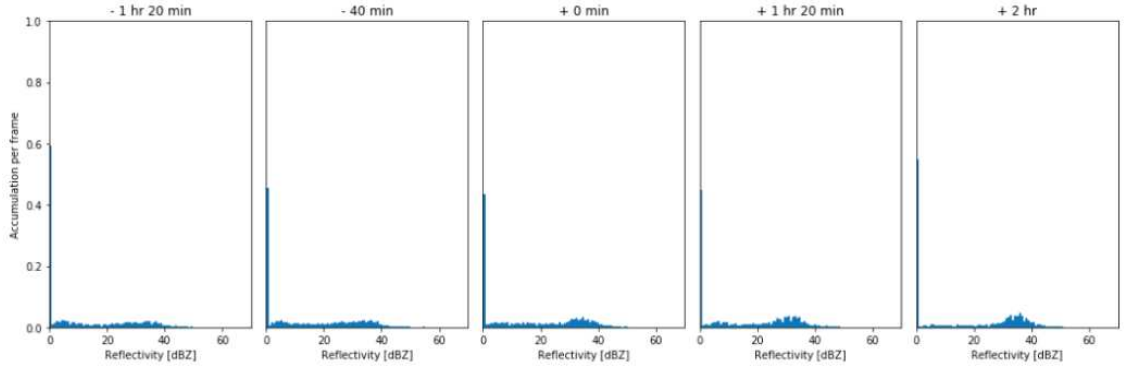
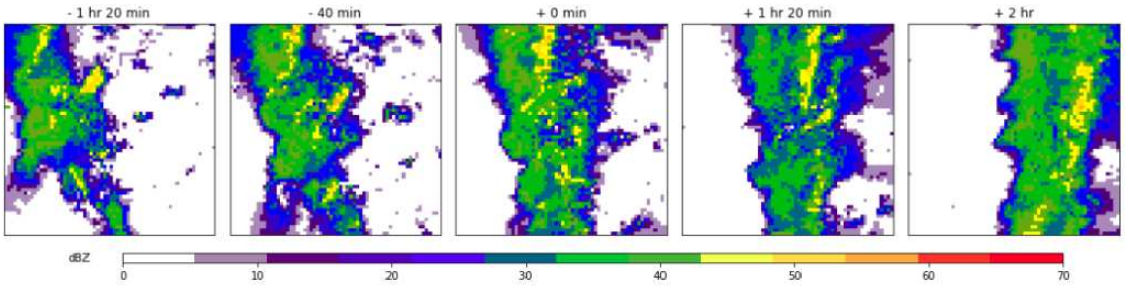


Figure A.3: Histogram of Z content on the training dataset.



(a)



(b)

Figure A.4: Histograms (a) and plots (b) of an event with high reflectivity values.

A.3 Testing datasets

Testing datasets used in the experiments.

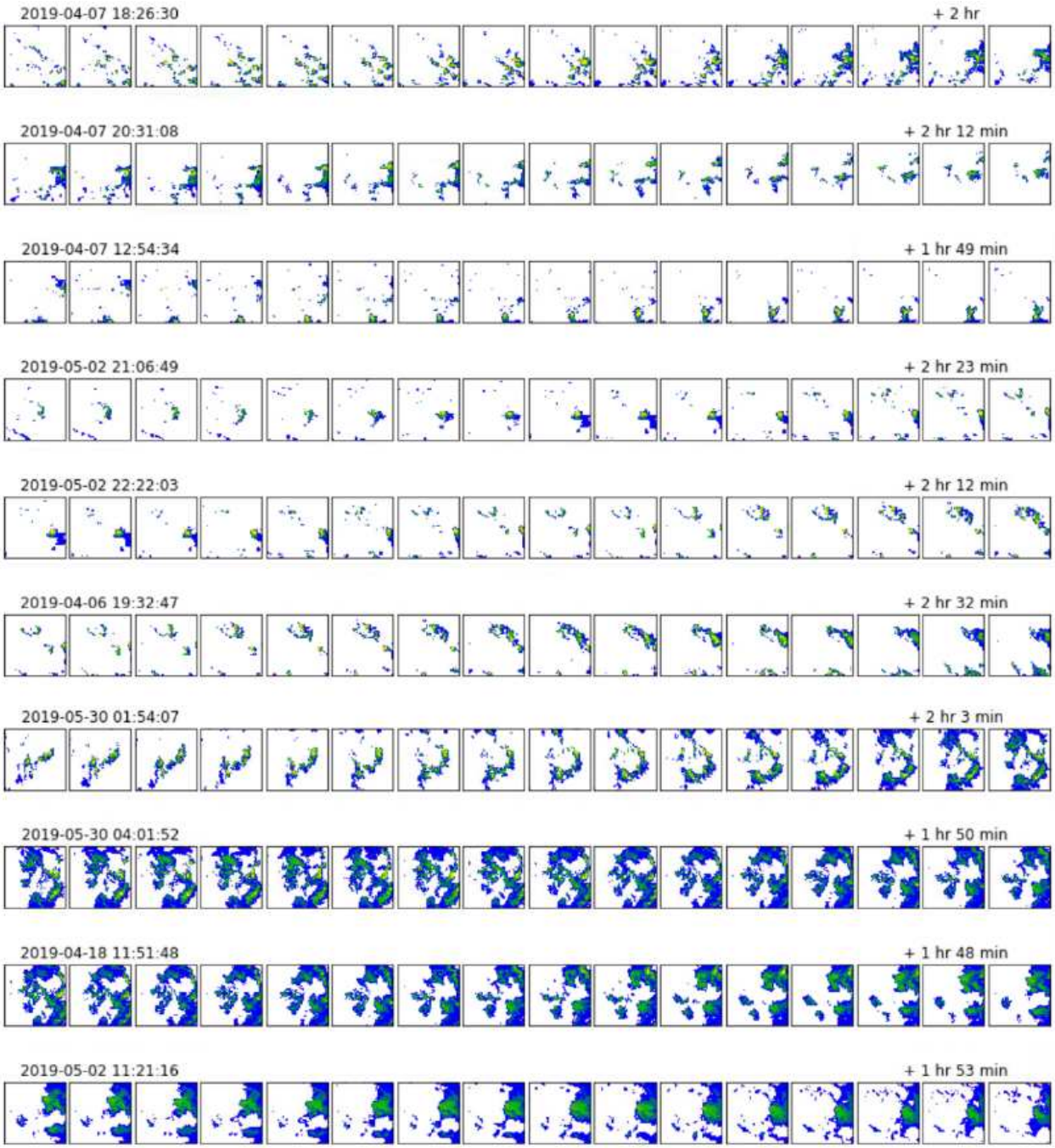


Figure A.5: Denver Spring 2017 selected events.

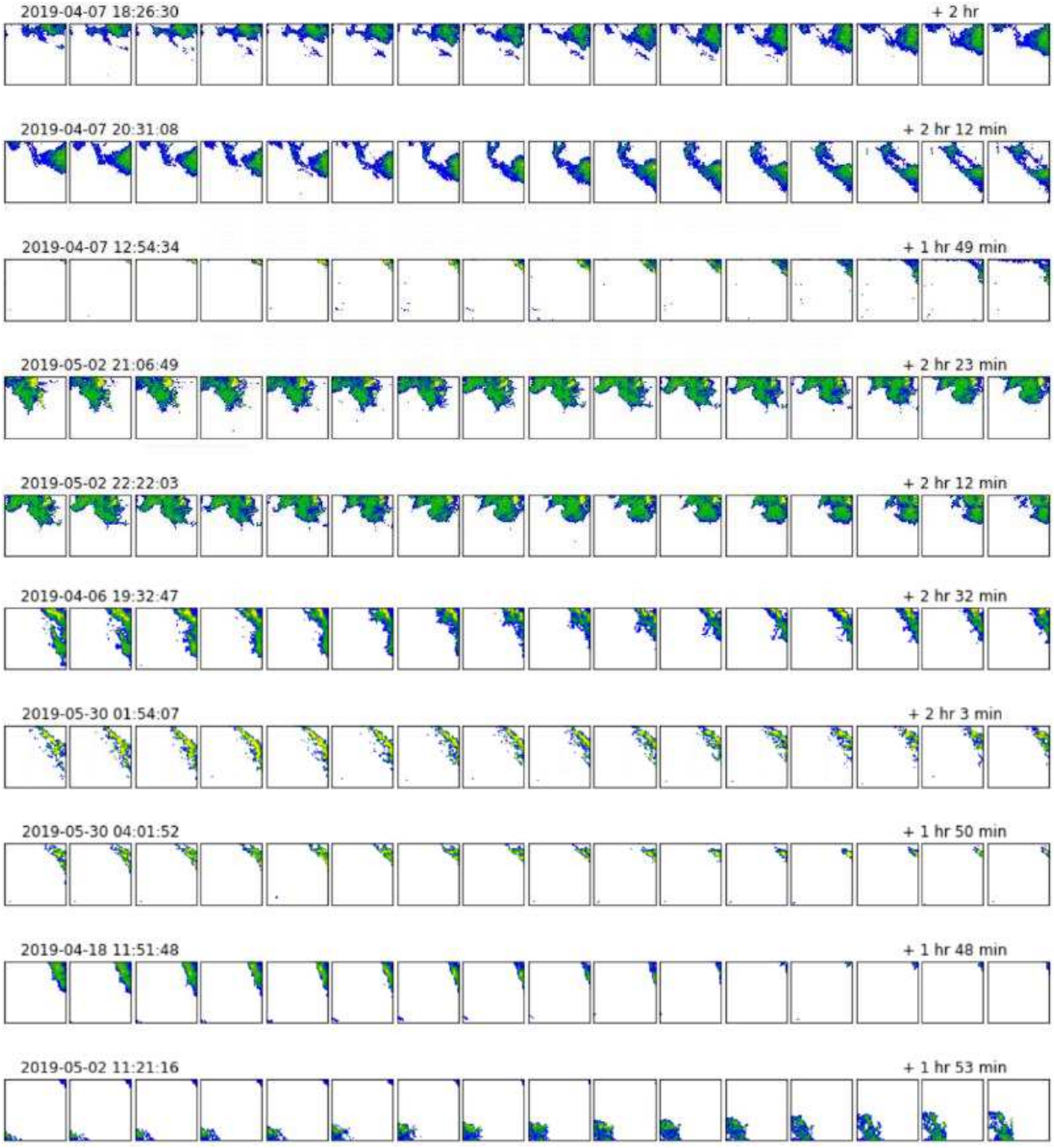
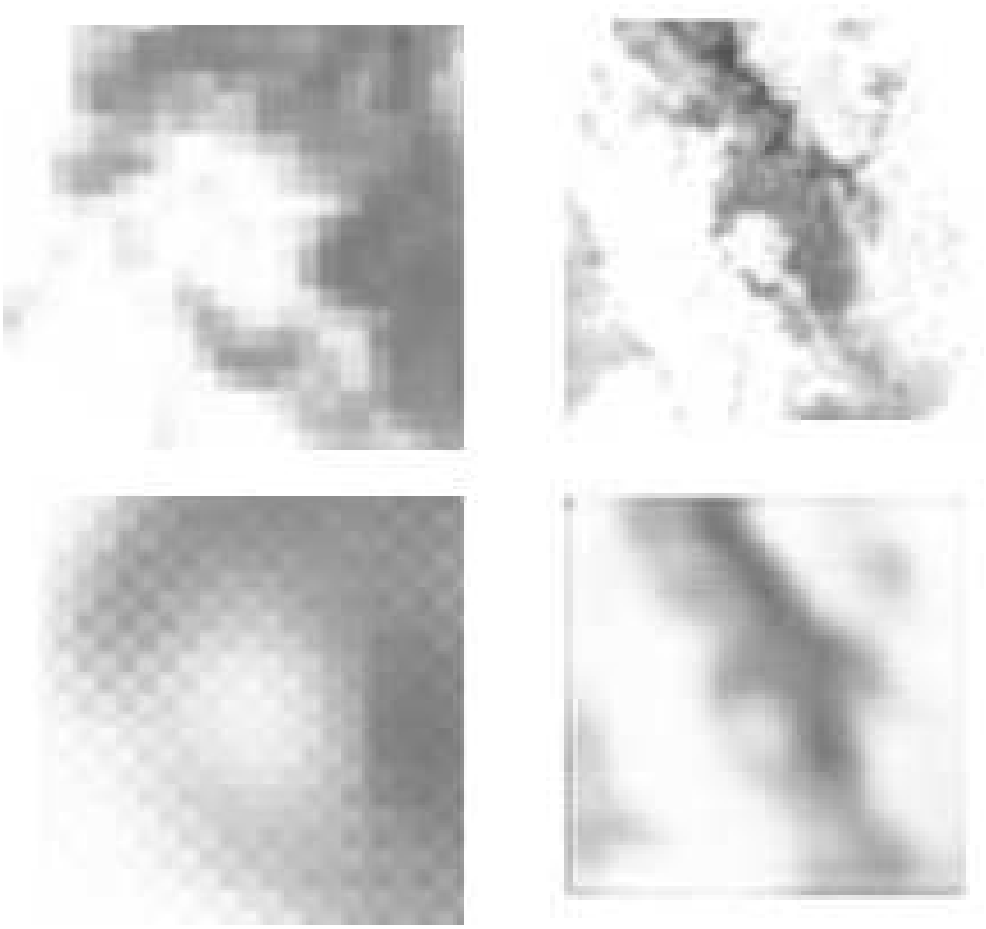


Figure A.6: Dallas Forth Worth Spring 2019 selected events.

A.4 On the generation of frames

Using the encoder-decoder architecture can arise problems on the upsampling part such as checkerboard artifacts [79], and edge artifacts. To minimize the first, the convolution strides were kept multiple of the kernels. For the second, the right padding was necessary, but still during the training phase is common to see how the edges are the last to show correct predictions. Figure A.7 illustrate both problems.



(a) **(b)**

Figure A.7: Example of (a) checkerboard artifacts, and (b) edge problem during first epochs of training. Top is observation and bottom prediction.

To evaluate if the already trained model suffers from edge problems the following analysis was done. Using the validation set with 134 events, the metrics were average sample-wise instead

of frame-wise and then average sample-wise. The result is that each pixel in each frame has the average metric for the location. Only MSE, ETS, and FAR are shown in Figure A.8. The results show no clear pattern that would indicate that in the edges is performing worst. The metrics would be expected to be homogeneous across each frame, but it is not due to the sample size. With different datasets, the pattern varies, which means that it is not performing better in the upper right corner as these results suggest.

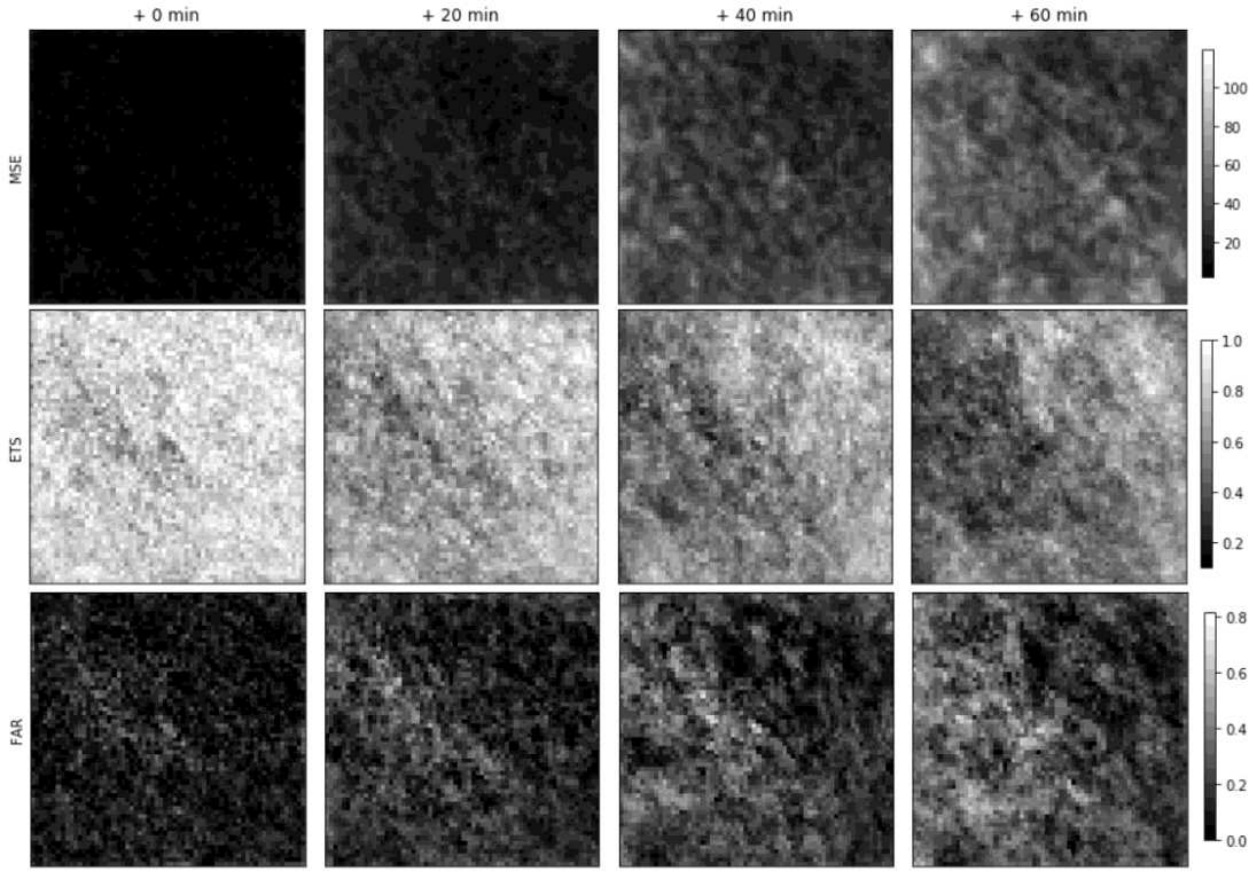


Figure A.8: Pixel-wise metrics for the edge problem detection.

In an attempt to measure the blurriness on the edges a Laplacian kernel was convolved (which estimate the second-order derivative of a matrix and it is interpreted as rapid intensity changes in an image) on each frame and then the standard deviation was computed on each pixel (with a filter size of 3x3). The higher the variance the less blurry an image is [80]. Because a kernel is

convolved twice this would definitely introduce artifacts on the edges as they are abrupt transitions, which interfere directly with the goal of the experiment. For that, different ways of extending the frames were tried and all yield similar results. As can be seen in Figure A.9 a comparison between the observation and the prediction was done. In the predictions the edges have higher values, which is the opposite of what was expected, meaning that it is less blurry than in the rest of the image. The lack of these edges on the observation frames means that it is not a consequence of the method implemented to detect blurriness. Therefore, despite that what it is being detected here is not actually blur on the predictions, it is certainly a type of artifact.

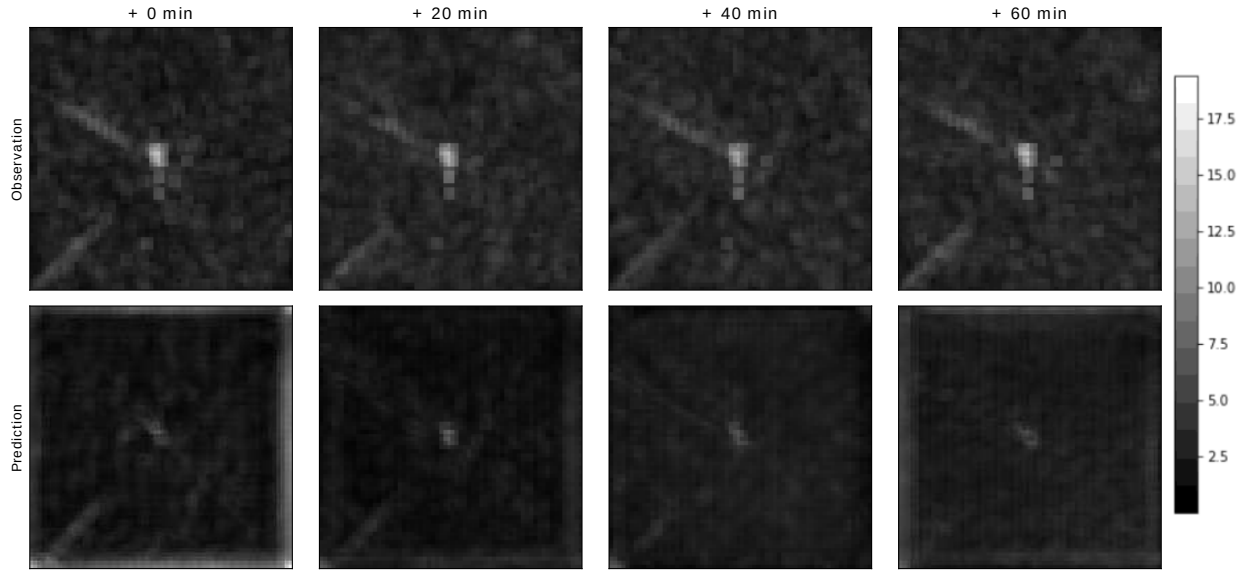


Figure A.9: Pixel-wise blur detection plots.

Two other 'anomalies' appeared on the blur analysis. One in the center with a circular shape and the other like rays from the center, which are stronger in the observations. The former is most likely the radar itself, and the later could be a calibration issue on the radar, as it seems to have a more abrupt change between two parts of the sweep.

A.5 KTH dataset

For potential users of the proposed model ResConv in different video prediction applications than weather nowcasting the results over KTH dataset [81] are shown in Figure A.10 and A.11.

Figure A.10: Prediction examples on KTH dataset.

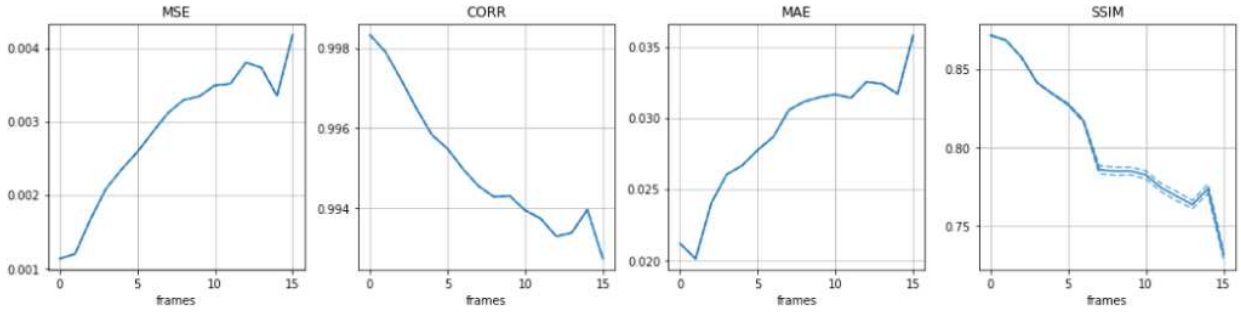


Figure A.11: Prediction metrics on KTH dataset.

A.6 Models' architectures

In here all the architectures are described in more detail. The terminology used here is:

- K and S for convolutional kernel and stride
- Output shape as (channels, frames, height, width)
- Skip connections are marked with an asterisk. Both outputs with same number of * are added.
- Initial states for recurrent layers are marked with a hyphen. Decoder (upsampling section) takes output state from layer with same number of -.

Table A.1: ResConv_32-16 model architecture

Layer	Parameters	Activation Normalization Regularization	Output shape	Skip Connections
Input			1 x 32 x 64 x 64	
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	128 x 16 x 32 x 32	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	128 x 16 x 32 x 32	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	128 x 16 x 32 x 32	*
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	256 x 8 x 16 x 16	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	256 x 8 x 16 x 16	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	256 x 8 x 16 x 16	**
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	384 x 4 x 8 x 8	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	384 x 4 x 8 x 8	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	384 x 4 x 8 x 8	***
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	512 x 2 x 4 x 4	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	512 x 2 x 4 x 4	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	512 x 2 x 4 x 4	
ConvT3D Upsampling	K=4x4x4 S=2x2x2	LeakyRelu	384 x 4 x 8 x 8	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	384 x 4 x 8 x 8	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	384 x 4 x 8 x 8	***
ConvT3D Upsampling	K=4x4x4 S=2x2x2	LeakyRelu	256 x 8 x 16 x 16	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	256 x 8 x 16 x 16	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	256 x 8 x 16 x 16	**
ConvT3D Upsampling	K=4x4x4 S=2x2x2	LeakyRelu	128 x 16 x 32 x 32	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	128 x 16 x 32 x 32	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	128 x 16 x 32 x 32	*
ConvT3D Upsampling	K=3x4x4 S=1x2x2	LeakyRelu	1 x 16 x 64 x 64	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	1 x 16 x 64 x 64	
ConvT3D	K=3x3x3 S=1x1x1	Sigmoid	1 x 16 x 64 x 64	

Table A.2: ResConv_32-32 model architecture

Layer	Parameters	Activation Normalization Regularization	Output shape	Skip Connections
Input			1 x 32 x 64 x 64	
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	128 x 16 x 32 x 32	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	128 x 16 x 32 x 32	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	128 x 16 x 32 x 32	*
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	256 x 8 x 16 x 16	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	256 x 8 x 16 x 16	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	256 x 8 x 16 x 16	**
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	384 x 4 x 8 x 8	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	384 x 4 x 8 x 8	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	384 x 4 x 8 x 8	***
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	512 x 2 x 4 x 4	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	512 x 2 x 4 x 4	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	512 x 2 x 4 x 4	****
Conv3D Downsampling	K=4x3x3 S=2x1x1	LeakyRelu	640 x 1 x 4 x 4	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	640 x 1 x 4 x 4	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	640 x 1 x 4 x 4	
ConvT3D Upsampling	K=4x3x3 S=2x1x1	LeakyRelu	512 x 2 x 4 x 4	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	512 x 2 x 4 x 4	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	512 x 2 x 4 x 4	****
ConvT3D Upsampling	K=4x4x4 S=2x2x2	LeakyRelu	384 x 4 x 8 x 8	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	384 x 4 x 8 x 8	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	384 x 4 x 8 x 8	***
ConvT3D Upsampling	K=4x4x4 S=2x2x2	LeakyRelu	256 x 8 x 16 x 16	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	256 x 8 x 16 x 16	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	256 x 8 x 16 x 16	**
ConvT3D Upsampling	K=4x4x4 S=2x2x2	LeakyRelu	128 x 16 x 32 x 32	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	128 x 16 x 32 x 32	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	128 x 16 x 32 x 32	*
ConvT3D Upsampling	K=4x4x4 S=2x2x2	LeakyRelu	1 x 32 x 64 x 64	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	1 x 32 x 64 x 64	
ConvT3D	K=3x3x3 S=1x1x1	Sigmoid	1 x 32 x 64 x 64	

Table A.3: ResConv_16-16 model architecture

Layer	Parameters	Activation Normalization Regularization	Output shape	Skip Connections
Input			1 x 16 x 64 x 64	
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	128 x 8 x 32 x 32	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	128 x 8 x 32 x 32	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	128 x 8 x 32 x 32	*
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	256 x 4 x 16 x 16	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	256 x 4 x 16 x 16	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	256 x 4 x 16 x 16	**
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	384 x 2 x 8 x 8	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	384 x 2 x 8 x 8	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	384 x 2 x 8 x 8	***
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	512 x 1 x 4 x 4	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	512 x 1 x 4 x 4	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	512 x 1 x 4 x 4	
ConvT3D Upsampling	K=4x4x4 S=2x2x2	LeakyRelu	384 x 2 x 8 x 8	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	384 x 2 x 8 x 8	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	384 x 2 x 8 x 8	***
ConvT3D Upsampling	K=4x4x4 S=2x2x2	LeakyRelu	256 x 4 x 16 x 16	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	256 x 4 x 16 x 16	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	256 x 4 x 16 x 16	**
ConvT3D Upsampling	K=4x4x4 S=2x2x2	LeakyRelu	128 x 8 x 32 x 32	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	128 x 8 x 32 x 32	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	128 x 8 x 32 x 32	*
ConvT3D Upsampling	K=4x4x4 S=2x2x2	LeakyRelu	1 x 16 x 64 x 64	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	1 x 16 x 64 x 64	
ConvT3D	K=3x3x3 S=1x1x1	Sigmoid	1 x 16 x 64 x 64	

Table A.4: ResConv_8-16 model architecture

Layer	Parameters	Activation Normalization Regularization	Output shape	Skip Connections
Input			1 x 8 x 64 x 64	*
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	128 x 4 x 32 x 32	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	128 x 4 x 32 x 32	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	128 x 4 x 32 x 32	**
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	256 x 2 x 16 x 16	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	256 x 2 x 16 x 16	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	256 x 2 x 16 x 16	***
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	384 x 1 x 8 x 8	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	384 x 1 x 8 x 8	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	384 x 1 x 8 x 8	****
Conv3D Downsampling	K=3x4x4 S=1x2x2	LeakyRelu	512 x 1 x 4 x 4	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	512 x 1 x 4 x 4	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	512 x 1 x 4 x 4	
ConvT3D Upsampling	K=3x4x4 S=1x2x2	LeakyRelu	384 x 1 x 8 x 8	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	384 x 1 x 8 x 8	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	384 x 1 x 8 x 8	****
ConvT3D Upsampling	K=4x4x4 S=2x2x2	LeakyRelu	256 x 2 x 16 x 16	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	256 x 2 x 16 x 16	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	256 x 2 x 16 x 16	***
ConvT3D Upsampling	K=4x4x4 S=2x2x2	LeakyRelu	128 x 4 x 32 x 32	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	128 x 4 x 32 x 32	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	128 x 4 x 32 x 32	**
ConvT3D Upsampling	K=4x4x4 S=2x2x2	LeakyRelu	128 x 8 x 64 x 64	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	128 x 8 x 64 x 64	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	128 x 8 x 64 x 64	*
ConvT3D Upsampling	K=4x3x3 S=2x1x1	LeakyRelu	1 x 16 x 64 x 64	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	1 x 16 x 64 x 64	
ConvT3D	K=3x3x3 S=1x1x1	Sigmoid	1 x 16 x 64 x 64	

Table A.5: ResConv_8-8 model architecture

Layer	Parameters	Activation Normalization Regularization	Output shape	Skip Connections
Input			1 x 8 x 64 x 64	
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	128 x 4 x 32 x 32	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	128 x 4 x 32 x 32	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	128 x 4 x 32 x 32	*
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	256 x 2 x 16 x 16	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	256 x 2 x 16 x 16	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	256 x 2 x 16 x 16	**
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	384 x 1 x 8 x 8	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	384 x 1 x 8 x 8	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	384 x 1 x 8 x 8	***
Conv3D Downsampling	K=3x4x4 S=1x2x2	LeakyRelu	512 x 1 x 4 x 4	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	512 x 1 x 4 x 4	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	512 x 1 x 4 x 4	
ConvT3D Upsampling	K=3x4x4 S=1x2x2	LeakyRelu	384 x 1 x 8 x 8	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	384 x 1 x 8 x 8	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	384 x 1 x 8 x 8	***
ConvT3D Upsampling	K=4x4x4 S=2x2x2	LeakyRelu	256 x 2 x 16 x 16	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	256 x 2 x 16 x 16	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	256 x 2 x 16 x 16	**
ConvT3D Upsampling	K=4x4x4 S=2x2x2	LeakyRelu	128 x 4 x 32 x 32	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	128 x 4 x 32 x 32	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	128 x 4 x 32 x 32	*
ConvT3D Upsampling	K=4x4x4 S=2x2x2	LeakyRelu	1 x 8 x 64 x 64	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	1 x 8 x 64 x 64	
ConvT3D	K=3x3x3 S=1x1x1	Sigmoid	1 x 8 x 64 x 64	

Table A.6: ResConv_4-16 model architecture

Layer	Parameters	Activation Normalization Regularization	Output shape	Skip Connections
Input			1 x 4 x 64 x 64	*
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	128 x 2 x 32 x 32	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	128 x 2 x 32 x 32	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	128 x 2 x 32 x 32	**
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	256 x 1 x 16 x 16	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	256 x 1 x 16 x 16	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	256 x 1 x 16 x 16	***
Conv3D Downsampling	K=3x4x4 S=1x2x2	LeakyRelu	384 x 1 x 8 x 8	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	384 x 1 x 8 x 8	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	384 x 1 x 8 x 8	****
Conv3D Downsampling	K=3x4x4 S=1x2x2	LeakyRelu	512 x 1 x 4 x 4	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	512 x 1 x 4 x 4	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	512 x 1 x 4 x 4	
ConvT3D Upsampling	K=3x4x4 S=1x2x2	LeakyRelu	384 x 1 x 8 x 8	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	384 x 1 x 8 x 8	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	384 x 1 x 8 x 8	****
ConvT3D Upsampling	K=3x4x4 S=1x2x2	LeakyRelu	256 x 1 x 16 x 16	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	256 x 1 x 16 x 16	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	256 x 1 x 16 x 16	***
ConvT3D Upsampling	K=4x4x4 S=2x2x2	LeakyRelu	128 x 2 x 32 x 32	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	128 x 2 x 32 x 32	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	128 x 2 x 32 x 32	**
ConvT3D Upsampling	K=4x4x4 S=2x2x2	LeakyRelu	128 x 4 x 64 x 64	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	128 x 4 x 64 x 64	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm	128 x 4 x 64 x 64	*
ConvT3D Upsampling	K=4x3x3 S=2x1x1	LeakyRelu	128 x 8 x 64 x 64	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	128 x 8 x 64 x 64	
ConvT3D	K=3x3x3 S=1x1x1	Sigmoid	128 x 8 x 64 x 64	
ConvT3D Upsampling	K=4x3x3 S=2x1x1	LeakyRelu	1 x 16 x 64 x 64	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	1 x 16 x 64 x 64	
ConvT3D	K=3x3x3 S=1x1x1	Sigmoid	1 x 16 x 64 x 64	

Table A.7: ResGRU model architecture

Layer	Parameters	Activation Normalization Regularization	Output shape	Skip Connections
Input			1 x 16 x 64 x 64	
Conv3D	K=3x3x3 S=1x1x1		1 x 16 x 64 x 64	*
convGRU	K=3x3		1 x 16 x 64 x 64	-
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	32 x 8 x 32 x 32	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	32 x 8 x 32 x 32	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm/Dropout	32 x 8 x 32 x 32	**
convGRU	K=3x3		32 x 8 x 32 x 32	- -
Conv3D Downsampling	K=4x4x4 S=2x2x2	LeakyRelu	64 x 4 x 16 x 16	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	64 x 4 x 16 x 16	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm/Dropout	64 x 4 x 16 x 16	***
convGRU	K=3x3		64 x 4 x 16 x 16	- - -
Conv3D Downsampling	K=3x4x4 S=1x2x2	LeakyRelu	96 x 2 x 8 x 8	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu	96 x 2 x 8 x 8	
Conv3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm/Dropout	96 x 2 x 8 x 8	****
convGRU	K=3x3		96 x 2 x 8 x 8	- - - -
ConvT3D Upsampling	K=3x4x4 S=1x2x2	LeakyRelu	96 x 2 x 8 x 8	
convGRU	K=3x3		96 x 2 x 8 x 8	****, - - - -
ConvT3D Upsampling	K=3x4x4 S=1x2x2	LeakyRelu	64 x 4 x 16 x 16	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	64 x 4 x 16 x 16	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm/Dropout	64 x 4 x 16 x 16	
convGRU	K=3x3		64 x 4 x 16 x 16	***, - - -
ConvT3D Upsampling	K=3x4x4 S=1x2x2	LeakyRelu	32 x 8 x 32 x 32	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	32 x 8 x 32 x 32	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm/Dropout	32 x 8 x 32 x 32	
convGRU	K=3x3		32 x 8 x 32 x 32	**, - -
ConvT3D Upsampling	K=4x4x4 S=2x2x2	LeakyRelu	1 x 16 x 64 x 64	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu	1 x 16 x 64 x 64	
ConvT3D	K=3x3x3 S=1x1x1	LeakyRelu/BatchNorm/Dropout	1 x 16 x 64 x 64	
convGRU	K=3x3		1 x 16 x 64 x 64	*, -
ConvT3D	K=3x3x3 S=1x1x1	Sigmoid	1 x 16 x 64 x 64	