UNIVERSITÀ DI TRENTO

Department of Information Engineering and Computer Science

Bachelor's Degree in
Computer Science

Final Dissertation

Comparative Analysis of Prediction Models for Short-Term Forecasting

Supervisor
Mauro Brunato

Co-supervisor
Gabriele Franch

Student
Marco Di Francesco

Academic year 2020/2021

Acknowledgements

Thanks to the Data Science for Health unit at Fondazione Bruno Kessler for giving me the opportunity to work on this project. Special thanks to Gabriele Franch guiding me in the whole process, and Max, Alberto and Chiara for supporting me, being patient, and answering all such kinds of mathematical and physical questions.

Thanks to my parents and my sister for the support in this trip, solving all types of problems also from a long distance.

Thanks to Alessandro, Izzo, Chini, Michael, la fisica and all other friends that supported me during this journey, giving me the joy to come everyday in Povo.

Thanks to my flatmates Daniele and Riccardo for the support, helping me make the right choice in all kinds of domains.

Thanks to Susanna for helping me be, positively or negatively, the person I'm right now.

Contents

Abstract

Due to global warming, we are living in an epoch where extreme events are becoming more and more likely. Excessive rainfall can act as a trigger for water-related hazards, which is especially true in an increasingly urbanized area with steep topography, such as the area of Trentino and Alto-Adige. When vulnerable objects become exposed to such hazards, risk can manifest in terms of property damage and loss of lives.

This work aims to find the model to best estimate and predict the extreme rainfall precipitation in the area of Trentino. This model is used to generate weather alerts related to a specific area, in such a way that precautions can be taken to avoid damages.

The key challenge in alerts generation is to create the right amount of alerts that are both going to not underestimate an extreme event, and at the same time do not overwhelm receivers of the alerts.

The target of this project is going to be the Italian Civil Protection Department. The model is going to support decision-making during severe weather, for example, to decide whether to interrupt a train line exposed to debris flows, as well as in the context of regulation of sewage systems during storm events. Future developments of the project are going to open these types of alerts to a broader user base for tourism and agriculture.

In this project, both mathematical methods and machine learning models were used to generate precipitation predictions. Two state-of-the-art machine learning models were trained on the same dataset, and the outcome is compared.

Results of this work comparing the Trajectory GRU machine learning model, and the traditional mathematical method S-PROG, show the mathematical method being outperformed by the machine learning model in almost all the cases considered. Moreover, a comparison between Trajectory GRU and the novel IDA-LSTM shows the former outperforming the new deep learning model.

In chapter 1 the concepts of forecasting, nowcasting, and methods used to gather meteorological data are presented. Following, a section about forecasting introduces the two main prediction methods used in this work.

In chapter 2 the dataset used for the comparison of the model is presented, underlining storage of the files, input and output of the model, split of the train and test sets, structure and scale of representation, and pre-processing methods with thresholding applied to the images. A section is going to present visually the dataset to understand the motivations behind the performances of the models.

In chapter 3 methods are discussed. First prediction models are presented in detail, following model evaluation techniques used for evaluating the output of the model are shown, then the toolkit for the implementation is explained.

In chapter 4 the results of the models are shown and compared. Results are shown for each model, followed by model comparison. This chapter further compares our dataset, using the best model found in the previous results, with another dataset and underlines the differences.

In chapter 5 the prediction of the models and the motivation behind the performances are discussed. The second part of the chapter discusses the possible motivations of the different performances of the model on the compared datasets. Future developments are then presented, with the possible implementation of another forecasting model and the motivation behind it.

1 Introduction

In this section general notions of meteorological forecasting are introduced. The chapter contains an introduction to nowcasting and signal reflection meaning, along with forecasting methods.

1.1 Forecasting

Forecasting refers to the practice of predicting events that will happen in the future considering past and present events. In this work, the subject of meteorological forecasting is considered, which is the subject that studies the condition of the atmosphere to predict its evolution in the future.

The time frame considered in the prediction can have great differences when it comes to forecasting. For this reason, the term Nowcasting for short-term prediction was created. This term changes the time frame depending on the subject.

1.1.1 Meteorological Nowcasting

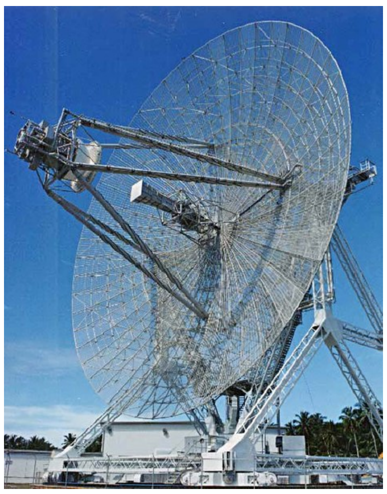
Meteorological Nowcasting refers to the prediction of the precipitation within a period of up to 2 hours [22].

A really short period of time was chosen to address the specific problem of detecting and localizing extreme events. This is different from traditional forecasting because we try to address the specific problem of localizing extreme precipitation and due to the inherent behavior of precipitation evolution we cannot have an accurate prediction of heavy precipitation with high accuracy over a long period.

1.1.2 Precipitation

Precipitation in meteorology is any product of the condensation of water vapor in the atmosphere, that under gravitational pull from the cloud falls on the earth. Precipitation occurs when a portion of the atmosphere has a concentration of water vapor, such that water condensates and precipitates.

The effects that water vapor visibly has can differentiate, for instance, fog and mist do not condense sufficiently enough to precipitate. These types of water vapor condensation are called colloids.



(a) Long-range radar used to track space objects

(b) Medium-range radar used to track aircrafts

Figure 1.1: Radar used to detect objects in the long distance [6]

1.1.3 Precipitation detection

There are multiple methods to detect the precipitation, divided into two categories, one is local sensing, so measuring the precipitation at a single point in space, and the other is remote sensing.

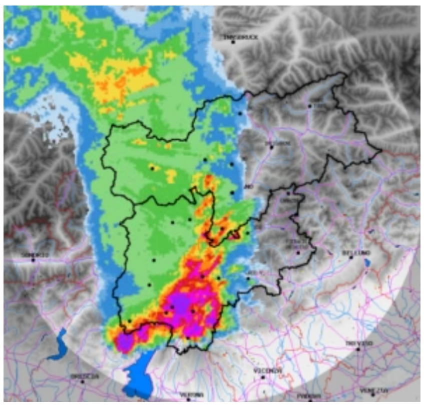
In this work, data coming from a Radar system is used. Radar stands for Radio Detection And Ranging system, a detection system that uses radio waves to determine the distance, angle, and velocity of objects.

Multiple types of radar are able to detect different objects depending on the electromagnetic waves produced. For instance, long-range radars are used to track space objects 1.1 (a) and others specializing in detecting aircrafts 1.1 (b).

The type of radar used in this work is a Weather Radar, as shown in picture 1.2 (b) a type of radar that emits a short pulse of energy, and if the pulse strikes an object the wave is scattered in all the directions, including the one of the radar. The amount of energy that the radar receives back is analyzed to understand if it came from an object, for instance, birds and bugs, or if it was coming from droplets' reflection.

The analysis of the radar is done using the Doppler effect to estimate not only the intensity but also the direction of the objects it detects, looking at the frequency of the wave it produced compared to the source wave sent.

When operating Weather Radars, the most interesting feature is the intensity of the precipitation, which is usually represented as a map where the value of each pixel corresponds to the intensity of the precipitation. Picture 1.2 (a) shows an example of the reflectivity map generated by a weather radar system, underlapped with the territorial image.


(a) Weather radar generated image

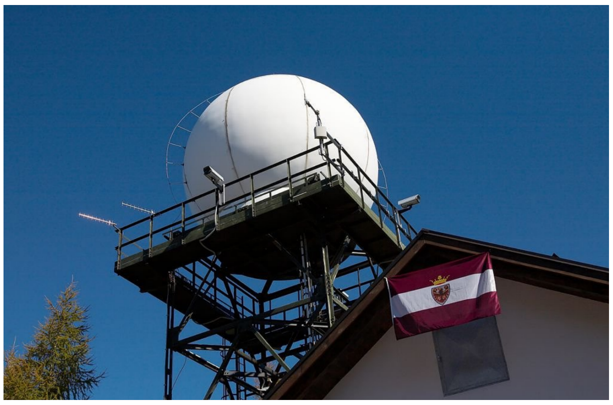

(b) Mt. Macaion weather radar

Figure 1.2: Weather Radar and generated reflectivity map [7]

1.1.4 Territory

The maps generated by radar systems may not reflect the real precipitation sensed on the ground. Images generated are 2D pictures containing in each pixel the precipitation value, for instance, a pixel can have a spatial resolution of 1 km, so covering an area of 1 km^2, and this may not reflect the reality due to territorial variation. As shown in picture 1.3 the grid spacing, when projected on steep terrain can have a huge difference in the area covered, covering a wider area of land depending on the slope of the orography.

These surface features can have an impact on the prediction when considering large territorial areas. Land features can change a lot over short distances, and given the size of the territory analyzed, the spatial resolution of the grid needs to be chosen according to the characteristics of the territory. The use of high-resolution grids helps to reduce orthographic projection problems and allows to represent of smaller-scale weather events, but on the other hand, high-resolution maps may be too computationally heavy to work with when making weather predictions.

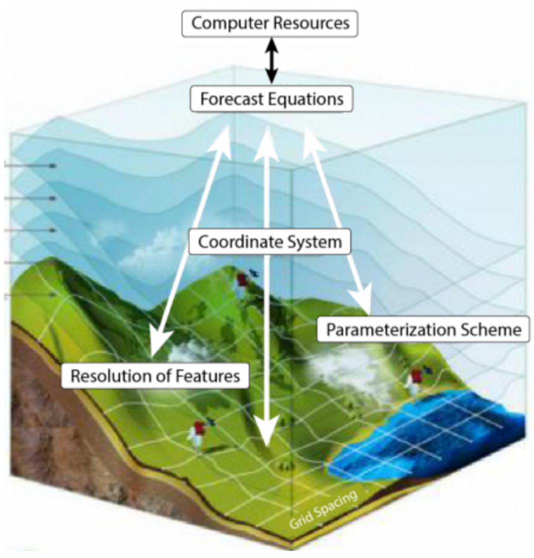


Figure 1.3: Projection of a regular grid on a complex orography [8]

1.1.5 Radar obstruction

Mountainous areas can also lead to the obstruction of radar sensing capabilities. If the beam is totally or partially blocked by a mountain, the precipitation sensed in the area behind the obstacle may represent a fraction or none of the real precipitation. The areas not sensed by the radar due to beam blockage are called shadow zones, and when a radar operates in mountainous regions it's common to encounter this situation.

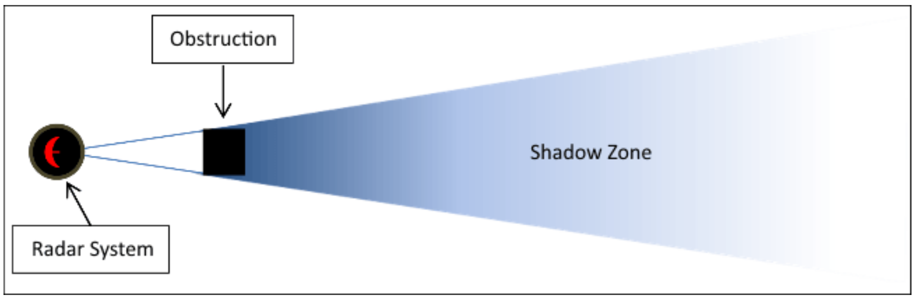


Figure 1.4: Radar obstruction [23]

This problem can be partially obviated by positioning the radar in the highest place possible, but due to environmental and accessibility constraints, this is not always feasible and these types of obstructions need to be considered when making forecasts.

1.2 Methods

In this section, precipitation forecasting methods are presented. The first type described is numerical, a mathematical method first introduced in 1980, the other is more recent, using machine learning to understand the evolution of precipitation.

1.2.1 Numerical Weather Prediction

The Numerical Weather Prediction (NWP) methods are mathematical systems that were first introduced by L.E. Richardson Et al. in 1920 but did not take hold until the 1950s when computer simulations allowed weather estimation in a feasible amount of time. These methods are based on the consideration of precipitation as fluids and using the equations of hydrodynamics by numerical methods to approximate the evolution of the precipitation.

The NWP models are used in the field of forecasting thanks to their ability to understand the physical evolution of the structures. The main difference from simple extrapolation methods is the ability to predict weather forecasts in a longer period of time, going from 4h up to several days.

1.2.2 Extrapolation approach

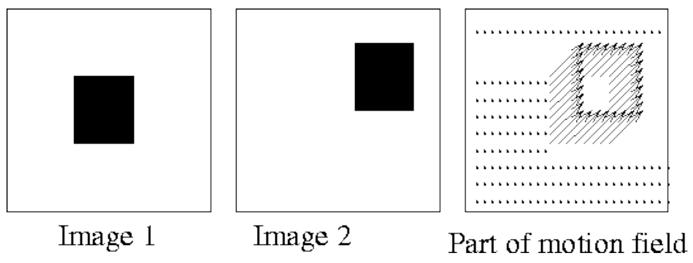
Nowcasting algorithms are based on the extrapolation of observed data, without considering the physics of the system. There are two main types of approaches, the first is the Eulerian approach, and the second is the Lagrangian approach.

In order to understand the latter approach, the concepts of motion field and optical flow are introduced.

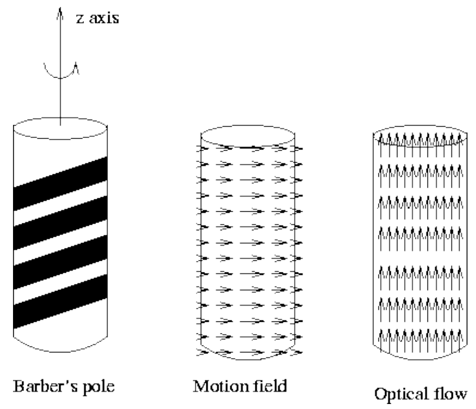
In the field of image processing, the motion field is the change in position of an object given a starting point. It can be described as given a starting point and a velocity vector, corresponding to the change of position of the object. In image 1.5 (a) in image 1 we have the starting point of the object, in image 2 the end position of the object, and in the last image the motion field that is created by the change of that object, with each arrow being the velocity vector.

On the other hand, the concept of optical flow is the detection of the apparent motion of brightness patterns in the image. This concept generally corresponds to the motion field, but not always.

In the image 1.5 (b) it's possible to see an example of a motion field compared to optical flow. Imagining the barber's pole, a sign rotating clockwise in the z-axis, the motion field is moving in the horizontal direction, given by the rotation of the pole, while the optical flow is moving in the vertical one, given by the stripes moving up.



(a) Motion field example



(b) Barber's pole motion field and optical flow

Figure 1.5: Motion and optical field example images [3]

The motion field explained above can be described as the image intensity I, a function of space x, y, and time t. We can describe the starting image $I(x, y, t)$ and the evolution with the following formula, where $u = dx/dt$ and $v = dy/dt$, are the vectors that describe where to move in the generated field.

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0$$

1.2.3 Machine learning

Machine learning is used in the field of forecasting to predict precipitation evolution. It uses Neural Networks to learn the parameters that characterize the evolution of the atmosphere.

The analysis used for the image is done by a Convolutional neural network (CNN), a type of feed-forward neural network that uses fully connected layers that learn the parameters of a kernel to extract the most useful information to reduce the number of features of the image.

The types of neural networks used are called Recurrent neural networks (RNN). They allow us to exhibit temporal dynamic behavior using an internal state (memory) to fit variable-length sequences of input. In the picture 1.6 we can see the general structure of an RNN. The main difference with a simple neural network is the ability to use the information of previous states, taking in input not only the data it has to analyze but also the temporal structure from the previous layer.

Theoretically, these types of networks should be able to learn well long-term dependencies, but

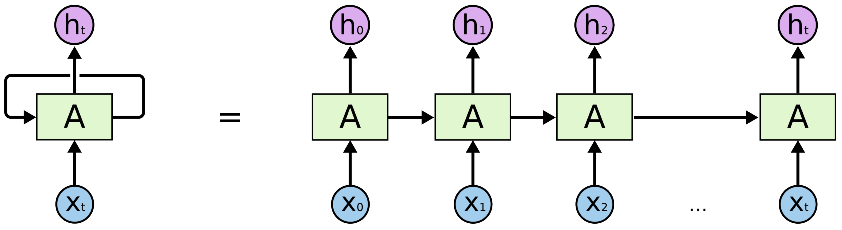


Figure 1.6: RNN unrolled to show input of each layer [18]

practically these networks struggle with this type of evolution. For this reason, Long Short Term Memory (LSTM) networks were created, allowing better learn long-term dependencies. These networks implement not only an input gate and an output gate, but also a forget gate. This gate is used to discover what details that need to be discarded from the block, effectively removing information that is not useful for long-term prediction.

2 Dataset

An important part of the project was understanding the data that needed to be fed to the model. The dataset used was TAASRAD19 by Franch Et Al. [11]. The dataset is a collection of data from the radar operated by Meteotrentino located on Monte Macaion, which generates a product every 5 minutes in the form of a radar precipitation map. This map is represented as an image with 480x480 pixels in size, where each pixel has a spatial resolution of 500 meters for a total of 240 km in diameter. This means that when the radar is fully operational, 288 images are generated every day.

The dataset used includes all images recorded from June 1, 2010, to December 31, 2019, for a total of 362,233 images.

2.1 Storage

The dataset is made out of 1,731 files in format HDF5, this format allows to manage, process, and store heterogeneous data and was built for fast I/O processing and storage.

Each file contains a contiguous block of 24 hours images, where each frame is taken every 5 minutes, each block contains 288 frames if the collection of the data was not interrupted. In the case the collection of data was not complete due to radar failure of maintenance, the number of frames per block is lower. An important thing to notice is that if the failure is contained in the middle of a 24-hours block, 2 or more independent blocks are created to avoid inconsistent groups of contiguous frames.

2.2 Split

For the machine learning task considered in this work, the dataset has been split into train and test sets. This division was made by taking into account the blocks from June 1, 2010, until the end of 2017 for the training, and from the beginning of 2018 until December 31, 2019, for the test.

These 2 sets split data into 193,611 frames for training and 168,622 frames for testing.

To benchmark the different nowcasting techniques, we use sequences of 25 frames split into 5 frames (25 minutes) given as input to the model and the following 20 (100 minutes) used to test the prediction of the precipitation evolution in the following instants of time. After each iteration, the window shifts by one frame, in such a way that each image is seen 5 times by the model, and used 20 times as a comparison.

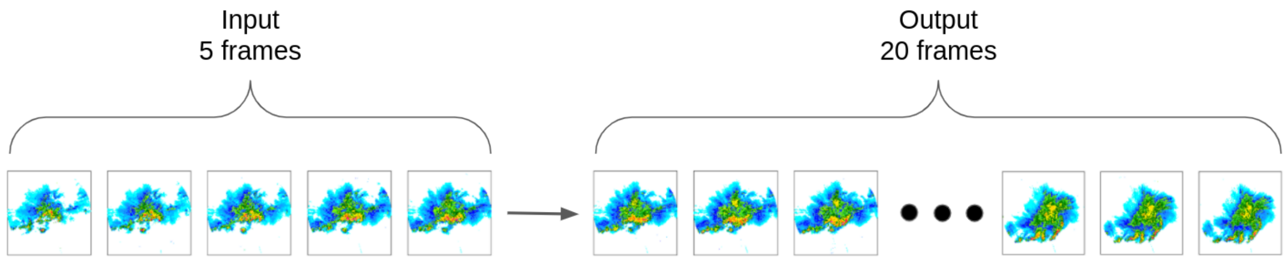


Figure 2.1: Input and output frames

2.3 Structure

When preprocessing the data a metadata file was created with different information about each block of images. This metadata is useful to apply filters and to avoid loading the whole dataset in memory.

Each block has the following information:

- **ID**: incremental identification number of the 24-hours block
- **Start datetime**: date and time of the first frame contained in the block
- **End datetime**: date and time of the last frames of the block
- **Run length**: number of contiguous frames in the block
- **Average reflectivity value**: average precipitation value of each pixel for all the images of the block
- **Tag**: type of precipitation based on a table containing the precipitation for that specific reflectivity range based on the average reflectivity value. There are 18 types of values and some examples are rain, snow, storm, and hail storm.

2.4 Scale of representation

The type of data we are using is Radar echo maps, where radar echo is the signal reflected off the precipitation by the radar. This type of data is collected using a radar, as explained in detail in section 1.1.3, that sends waves that when encountering precipitation droplets are bounced back, effectively creating a precipitation map.

What the radar is reflecting is known as Hydrometeor, any water or ice particles that have formed in the atmosphere as a result of condensation or sublimation. Some examples are clouds, rain, and snow.

The intensity of the precipitation is measured by the radar in the reflectivity factor Z, which is dependent on the number and size of hydrometeors. In the case of meteorology, it represents the size distribution of the hydrometeors in the unit volume considered. In our case, the reflectivity is expressed in mm^6 per cube meter, equivalent to μm^3.

Since the reflexivity factor, Z covers a wide range of values, from very weak like drizzle to very strong signals like hail, it was more convenient to express it in decibel, using the logarithmic decibel scale. It is calculated taking into consideration the reflectivity factor Z divided by the base value Z_0 that in our case corresponds to a 1 mm drop in a volume of a meter cube.

$$L_Z = 10 \log_{10} \frac{Z}{Z_0} dBZ$$

The scale allows us to divide and label the reflectivity value on a continuous scale as shown in table 2.1.

The representation of this reflectivity value in decibel can be done using a continuous color palette. The figure 2.2 represents the precipitation with a logarithmic scale with values ranging from 0 to 160 mm/h.

L_Z *(dBZ)*	*R (mm/h)*	**Intensity**
5	0.07	Hardly noticeable
10	0.15	Light mist
15	0.3	Mist
20	0.6	Very light rain
25	1.3	Light rain
30	2.7	Light to moderate rain
40	11	Moderate rain
50	48	Heavy rain
60	205	Hail

Table 2.1: Reflectivity in dBZ and rain rate comparison [26]

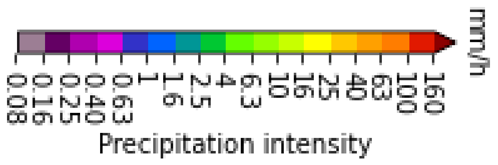


Figure 2.2: Precipitation intensity scale (mm/h)

Given that the input of the models was normalized from 0 to 1 on a linear scale, all values were divided by the all-time maximum reflectivity value recorded by the radar (in our case was 52.5 dBZ) minus the minimum (for us 0 dBZ).

$$L_{ZNorm} = \frac{L_Z}{(max(L_Z) - min(L_Z))}$$

2.5 Pre-preprocessing

Because of the computational cost that deep learning techniques require to work, the training and test sets were reduced.

The region of Trentino, the area analyzed in this dataset, has an average precipitation of only 82 days per year on average per each possible location of the region [2]. For this reason, we had to discard sunny days and the ones with light precipitation.

In the histogram 2.3 the X-axis represents the average reflectivity value in a block for each pixel and in the Y-axis the number of times that block had a specific amount of rain. From this plot, we can see how often we have days with little precipitation and how rare precipitation that lasts the whole day is. Because of this, we decided to exclude blocks with an average reflectivity value per pixel less than 1.5 dBZ, corresponding to short, isolated, and light precipitation in the region.

The filtered dataset used by the model consisted of 86,700 sequences for the training dataset and 96,494 for the test, which corresponds to around 50.5% of the original sequences.

2.6 Rain intensity and Thresholding

The goal of the project was to identify and predict extreme and localized harmful events. As pointed out by Pan et al. [19] thresholding data can achieve an increase in performance due to the low precipitation values not giving a considerable contribution to the forecast, and only raising complexity when learning the precipitation structure. For this reason, a pre-processing procedure of the dataset has been prepared, where the least intense rainfall is identified and removed, avoiding the model to predict values of rain that would not be useful for our purpose.

The minimum rainfall threshold was applied to each frame, setting values below a certain threshold to 0. An example is shown in the plot 2.3, where the left plot contains the full image with all the values and the image in the right the values below 0.3 mm/h, corresponding to 15 dBZ, were cut. This allowed the model to discard the area of precipitation not useful for the prediction, effectively cutting

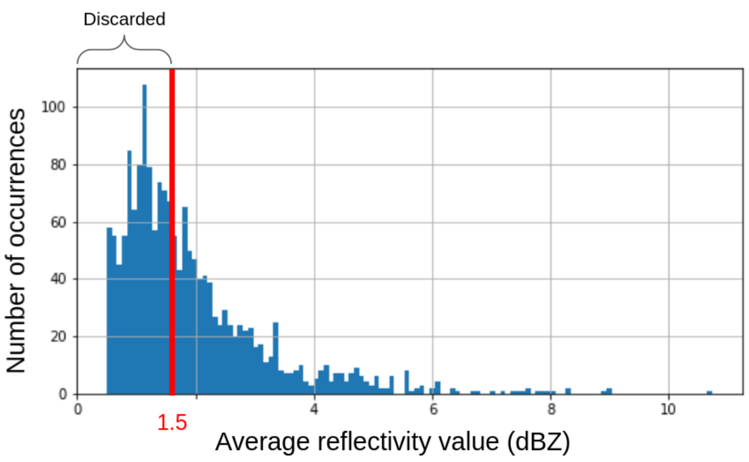


Figure 2.3: Histogram with average reflectivity value per block

off the lowest radar rainfall values, focusing the model loss on the highest precipitation values.

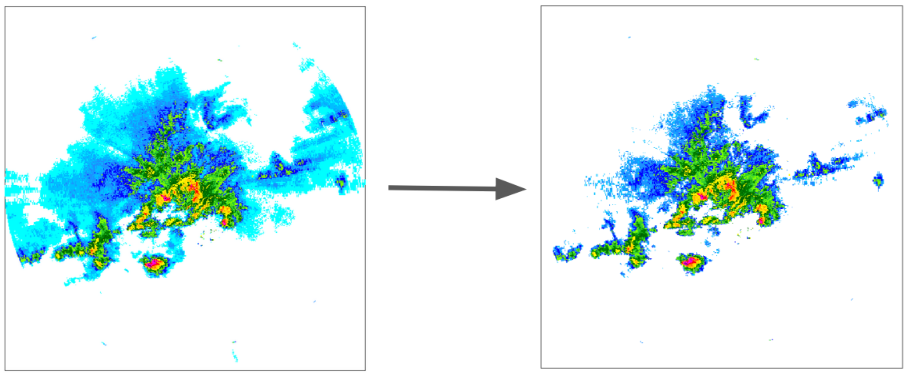


Figure 2.4: Block thresholding of a frame cutting values below 0.3 dBZ

The choice of the thresholds was made considering a trade-off between the amount of information that the model should not predict, for instance very low precipitation values like drizzle, and at the same time not discarding information that could be useful to the model to detect and predict intense precipitation.

We accordingly trained the model on the following thresholds of: 0.01 mm, 0.03 mm/h, 0.1 mm, and 0.3 mm; corresponding to values that represent hardly noticeable mist to a light fog.

2.7 Visual inspection

The analysis of the images was useful to understand how the dataset is composed, by visualizing the images using a color scale that reflects the intensity of the precipitation, it is easy to understand how the precipitation evolves in space and time.

The figure 2.5 is an example of a sequence of frames of the dataset in temporal order (5 minutes apart): for this example, we used the flood of July 3, 2018, that shows the evolution of intense precipitation using the color scale explained above.

In a visual inspection, it's possible to notice a few things. First, we can see that the pictures have a circular shape given by the type of radar we are considering, and we can notice a beam wrongly classified as precipitation at the bottom of the picture that is going to be removed by pixel-wise post-processing.

Another important thing to notice in these pictures is that despite the precipitation being spread across a big area (colored in violet), the area of strong precipitation (from green to yellow) is very small. Indeed the hail storm in these pictures is concentrated on a very small area that is corresponding with the outcome this flood had on the area affected.

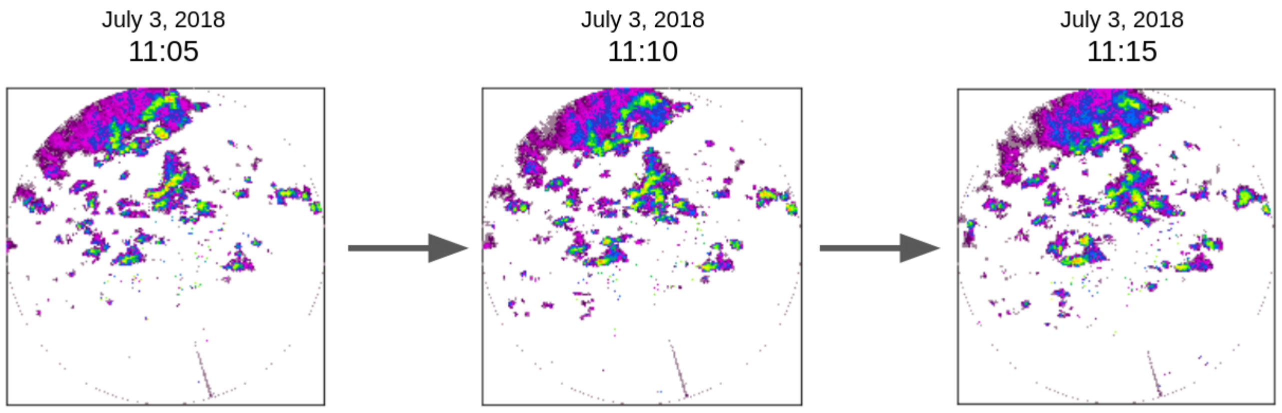


Figure 2.5: Moena flood precipitation event

A third thing we should notice is the speed at which the precipitation is moving. Indeed the precipitation in the picture moves at very different paces throughout the region, for instance in the very right corner of the picture we can see that the clouds stagnate, meanwhile in the left corner of the picture the precipitation moves quickly to the right.

These observations are really important to understand the results of the prediction models, in order not to look at the wrong parameters when comparing and evaluating models.

3 Methods

In this chapter, three main methods used in the project are explained. First, explanation of the prediction model and the motivations for using them, second, evaluation methods used to compare these models, and third how they were implemented.

3.1 Prediction models

In this project, different models were tested employing both mathematical and statistical techniques. We started with a state-of-the-art mathematical model that uses the optical flow of the picture to make the prediction, then we trained Trajectory GRU, a well known deep learning model that uses a recurrent neural network to make accurate predictions, and finally a novel model called IDA-LSTM that implements a mechanism to improve performances upon simple convolutional recurrent neural networks. The former two models were tested fine-tuning the parameters to get optimal results from each one of them.

3.1.1 Optical flow method

Spectral Prognosis (S-PROG) [24] is a mathematical model proposed by Seed et al. that makes a prediction using an extrapolation approach with the optical flow method.

This model works by computing the advection matrix using the semi-lagrangian extrapolation method. This method allows using an autoregressive model to approximate the Lagrangian space that describes the evolution of the precipitation, making the decomposition of the field into Fourier components, and separating the motion of the field from that due to the temporal evolution of the field itself. These two lagrangian multipliers allow describing the evolution using a lightweight computation.

S-PROG moreover uses the transformation of the field using multiple Fourier transformations to get multiple spatial representations of the precipitation that are used to get structures of different sizes. Picture 3.1 contains an example of precipitation decomposed in different levels, where the scale of color used has white as the lowest value and red as the highest. This is useful to get at the same time small structures like localized cyclones, and bigger thunderstorms.

The advection matrix computed by this model maps each point to a pixel containing the direction

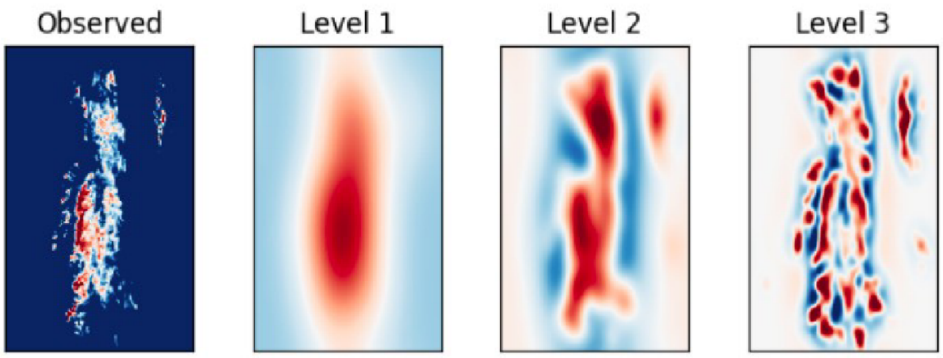


Figure 3.1: Results of different fourier transformation decomposition [9]

where the precipitation should be moving to.

Image 3.2 shows an example of how the evolution of this extrapolation method works. This example uses the same matrix on all 3 frames and assigns the intensity value of each pixel by looking at the values of its upwind neighbors.

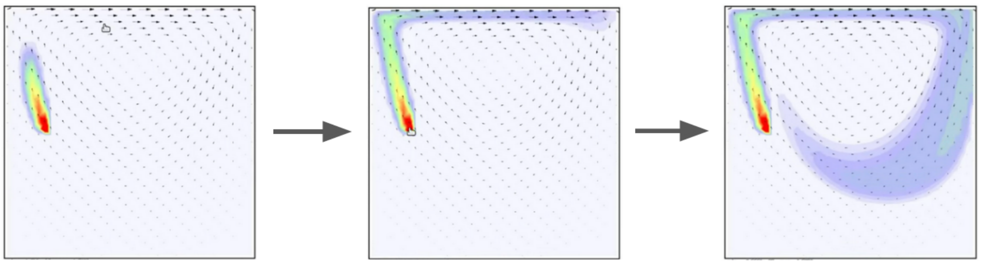


Figure 3.2: Optical flow example using semi lagrangian extrapolation method [14]

The configuration of the implementation of this model, taken from [11], is the same settings of the other machine learning models. It uses the first 5 frames of the precipitation as input, predicting the following 20 frames. In this model, the picture of the last frame of the input frames is fed to the first matrix, which will output the picture that will be fed to the next matrix up to the last frame.

3.1.2 Trajectory GRU

Trajectory GRU is a model that uses a modification of Convolutional LSTM that actively learns location-variant structure for recurrent locations.

The implementation of this model is discussed in the publication, Deep Learning for Precipitation Nowcasting A Benchmark and A New Model [25]. This model was trained using metrics that can be found in the publication Precipitation Nowcasting with Orographic Enhanced Stacked Generalization: Improving Deep Learning Predictions on Extreme Events [11].

This model works by using an Encoding-Forecasting structure, where, given a sequence of an arbitrary number of frames, we use many recurrent neural networks to learn different spatiotemporal representations.

In the picture 3.3 we can see the structure used by this model, the left side representing the encoder and on the left side the forecaster.

The Encoder model takes as input an arbitrary number of frames ($I1$, $I2$), makes a simple convolution in the first layer, and feeds the output into the first recurrent neural networks (RNN) where is then downsampled using a convolution with stride and fed to a second RNN, where this process is repeated for a second time.

The Forecaster model takes as input the prediction of the memory of the last RNN of the encoder, it then upsamples it using deconvolution with stride and feeds it to the next RNN where this process is repeated for a second time. The output is then fed to the last RNN that makes the prediction that is then deconvolved for the last time.

Each RNN is a Gated Recurrent Unit (GRU), a network consisting of a cell, an input gate, and an output gate. In our implementation, the GRU takes as input the processed picture and the memory of the RNN of the previous layer (from the output gate), except for the first layer where zeros are fed as input.

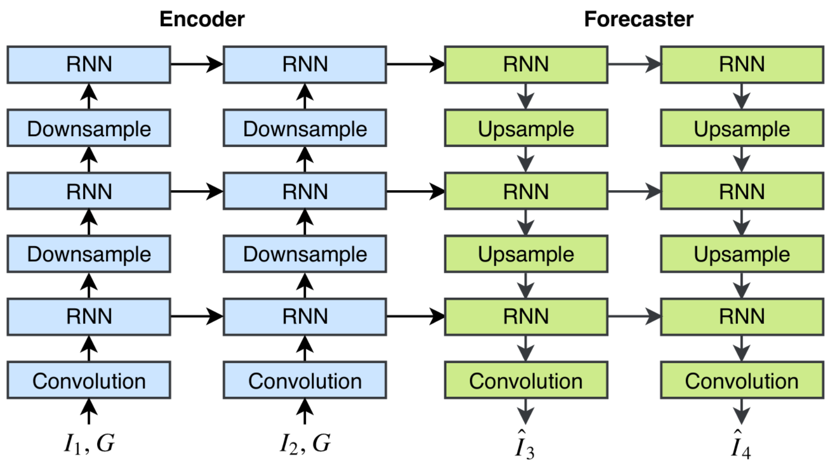


Figure 3.3: Encoder-Forecaster structure used in TrajGRU [25]

Trajectory GRU proposes a variation from the traditional Convolutional RNN. In the former networks, the convolution operation applies a location-invariant filter of the input, meaning that the weights are fixed for all the locations. This model instead proposes a Trajectory RNN where recurrent connections are dynamically determined, as shown on the right side of picture 3.4.

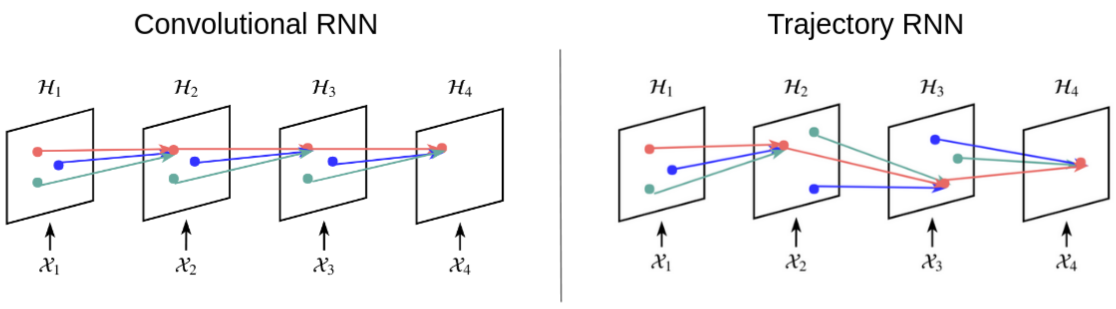


Figure 3.4: Convolutional RNN and Trajectory RNN connections comparison [25]

The loss function used by Trajectory GRU is Balanced Mean Squared Error (B-SME), a function that assigns more weights to heavier rainfalls in the calculation of MSE and MAE.

As shown in the histogram 2.3 the intensity distribution of the rain is concentrated around one millimeter, with very little precipitation exceeding four millimeters. For this reason, the authors decided to consider thresholds and assign a weight to them proportional to the class imbalance as shown in table 3.1.

Considering w as our weight, x as our predicted value, and $\hat{x}$ out target value, the balance MSE and MAE are calculated in the following way.

$$B - MSE = \frac{1}{n} * \sum_{n=1}^{20} \sum_{i=1}^{480} \sum_{j=1}^{480} w_{n,i,j}(x_{n,i,j} - \hat{x}_{n,i,j})^2$$

$$B - MAE = \frac{1}{n} * \sum_{n=1}^{20} \sum_{i=1}^{480} \sum_{j=1}^{480} w_{n,i,j}|x_{n,i,j} - \hat{x}_{n,i,j}|$$

Rain Rate (mm/h)	Proportion (%)	Weight
$0 \leq x < 0.5$	90.25	1
$0.5 \leq x < 2$	4.38	1
$2 \leq x < 5$	2.46	2
$5 \leq x < 10$	1.35	5
$10 \leq x < 30$	1.14	10
$30 \leq x$	0.42	30

Table 3.1: Rain Rate proportion with assigned weight

3.1.3 IDA-LSTM

IDA-LSTM is a model proposed by Luo et al. [16] that tries to fix a problem that S-PROG, Trajectory GRU, and in general simple optical flow and Convolutional RNN have, that is the underestimation of the high echo value parts. As shown in the picture 3.5 extreme values in Trajectory GRU tend to diminish over time, leading to a prediction that is not going to be useful for our purposes, that is to predict extreme events.

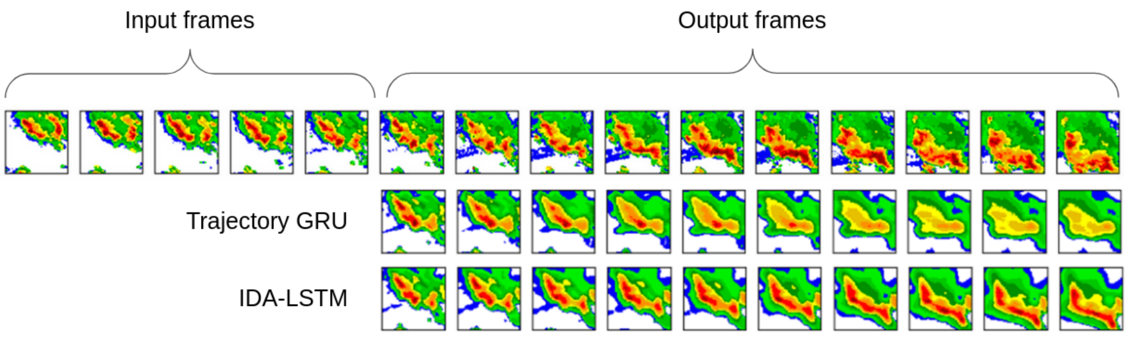


Figure 3.5: Trajectory GRU and IDA-LSTM comparison [16]

The model is built on the Convolutional RNN and implements the Interaction mechanism. This mechanism tries to address the problem of simple convolutional RNN like Trajectory GRU where input and hidden state make two independent convolutions. In the picture 3.6 we can see how the original input X and the hidden state h do not get just one convolution before being fed to the RNN unit, but a convolution in of both is made and summed, and a rectified linear threshold unit (ReLU) is appended. This can be repeated multiple times and our implementation repeated it for three.

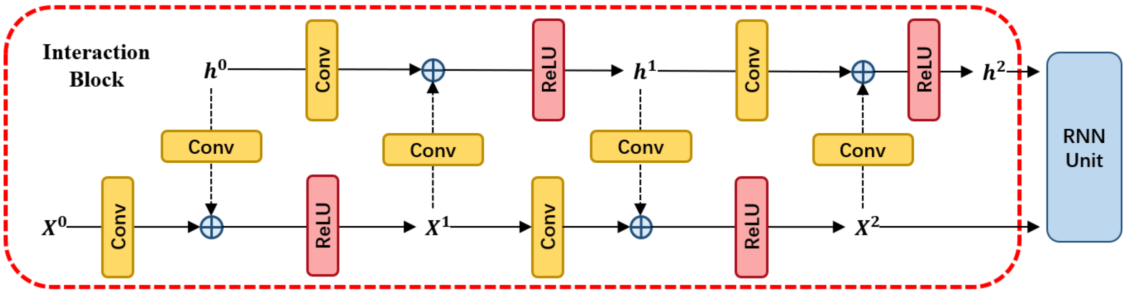


Figure 3.6: Interaction Framework visualization [16]

3.2 Model evaluation

Each model was trained to get the same output. The project aimed to have an output picture with a resolution of 480x480 pixels, 20 frames of prediction with the same timeframe of 5 minutes between each frame.

3.2.1 Output transformation

The values of the model, as explained in chapter 2.4, has a normalized value from 0 to 1. In order to be evaluated, values were transformed from reflectivity value (dBZ) to rain rate (mm/h) reversing the formula explained in chapter 2.4.

Each picture generated by the models had to be compared with the ground truth, and multiple methods were used to estimate the quality of the prediction.

3.2.2 Contingency table

When comparing two pictures, pixel values corresponding to the same position of the picture are taken into consideration and contingency tables are calculated.

A contingency table in the field of weather forecasting is a 2x2 matrix counting the number of pixels categorized as hit, miss, and false alarm. These categories are assigned considering a threshold, where the value is considered a ”Yes”, if it exceeds this threshold, otherwise is ”No”. The table compares each pixel of the observed forecast and the predicted forecast, and if both agree they are considered a Hit, meanwhile if the ground truth marks detect it and the forecast doesn't is considered as a Miss, and conversely if the ground truth doesn't and the prediction does is considered as a False alarm.

Contingency Table		**Event Observed**	
		Yes	No
Event Forecast	Yes	Hit	False alarm
	No	Miss	True Negative

Table 3.2: Contingency table

3.2.3 Evaluation scores

Once we have the contingency table with their categories, it's possible to introduce scores that are used to evaluate the quality of the prediction.

False Alarm Rate (FAR) is a score used to understand the number of pixels declared as a false alarm. It is calculated taking into consideration only the false alarms and hits of the prediction, not considering misses.

$$FAR = \frac{false\ alarms}{hits + false\ alarms}$$

Probability Of Detection (POD) is the parameter that calculates how many pixels have been correctly predicted considering the number of missing predictions.

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

Critical Success Index (CSI) is a metric that considers all three categories. It is calculated considering not only the number of pixels that were not marked as correct but also the number of pixels that were incorrectly predicted.

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

3.2.4 Plotting scores

Visual comparison of our results was done using the Roebber plot, a plot created by Roebber (2009) [4] that allows visualizing critical success index compared with the probability of detection and false alarm rate.

This plot has in the x-axis the Success Rate, calculated as $1 - (FalseAlarmRate)$, and in the y-axis, we have the Probability Of Detection. As shown in the left picture of 3.7 underlined in yellow we have the lines representing the critical success index, with values starting from 0 in the left bottom corner up to 1 in the top right. This allows us to draw oblique lines that are going to indicate the bias we have in the model. In the right picture of 3.7 we can see the yellow line underlining the perfect ratio between the probability of detection and success ratio and in red and blue the areas where the precipitation is respectively over predicted, with a high bias, and under-predicted, with a low bias.

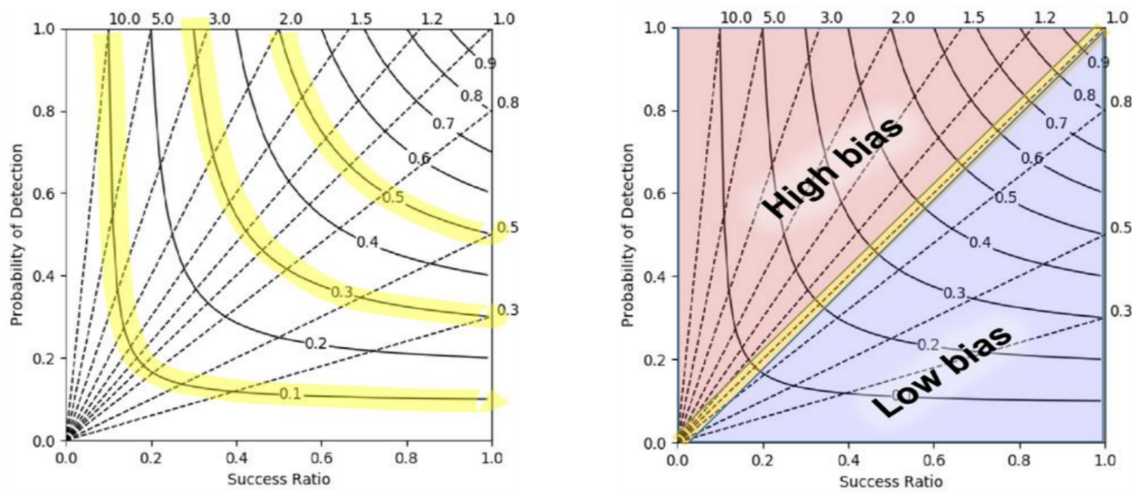


Figure 3.7: Roebber plot examples [10]

3.3 Implementation

Python programming language was used for the whole project thanks to its flexibility for the implementation of neural networks and data visualization.

Multiple libraries were used to accomplish this job, and they can be categorized under data analysis, dataset generation, neural network, and visualization libraries.

3.3.1 Data analysis

The data analysis was done using the Pandas framework [17] this framework uses the Numpy library [13] and exposes simple APIs to manipulate bidimensional arrays. It was used to analyze and manipulate the dataset's metadata as shown in sections 2.5 and 2.6.

3.3.2 Dataset preprocessing

PyTorch framework [20] was used for the dataset generation. More in particular two modules were used, one is Dataset, a submodule used to list the dataset files to create an iterable object, and the other is Dataloader a submodule used to implement loading of dataset blocks in memory.

These submodules together allow to map and load memory blocks that are given in input to the models asynchronously. It does this using python multithreading functionality, allowing the possibility to make image preprocessing in CPU, while the GPU is working on other tasks. This allows the model to train faster because they are not bottlenecked by the disk read time and preprocessing load time.

The creation of this dataset loader was an important part of the project, which drastically decreased GPU idle time, that is the timeframe between when a memory block is read from memory and when the GPU starts computing.

The PyTorch data loader furthermore allowed us to implement Python generators instead of simple iterators. This functionality was particularly useful given the large quantity of data our dataset had. Normal iterators would have allowed us to iterate the entire dataset, consisting of 86,700 blocks, for a fixed number of epochs, in our case either 2 or 3. On the other hand, this implementation allowed us to choose a fixed number of iterations, in our case 200,000, useful to make a fair comparison with other state-of-the-art models.

3.3.3 Model implementation

The pySTEPS library [21] implements mathematical models for precipitation nowcasting. It provides modular frameworks for nowcasting and stochastic space-time simulation of precipitation. This python module played an important role in the project because it implemented both the mathematical method, explained in 3.1.1, the evaluation metrics (3.2.3) and wrapper for the visualization function Matplotlib for maps visualization shown in 2.6.

Pytorch was used to implement the deep learning models Trajectory GRU and IDA-LSTM. These models were implemented using tensors, multi-dimensional matrices containing 32-bits floating-point elements, that were loaded and processed in the GPU in order to speed up the training times. This library furthermore allowed the parallelization between multiple GPUs, splitting the batches, to process multiple dataset blocks at the same time in different GPUs.

4 Results

In this chapter, the implementation and results of the models are going to be discussed. The first model analyzed is the IDA-LSTM model, where the input of the model was adapted to fit our dataset, and two configurations are proposed and discussed. The second model is Trajectory GRU, a model that is trained in four different configurations and tested two times for each, to get the best results fitting our project's requirements. The third model is used as a base comparison of the Trajectory GRU model, and it is going to be compared with the same parameters we used for S-PROG, in order to get a general idea of the performances of a mathematical model when compared to neural networks.

4.1 IDA-LSTM

The model was tested using different configurations. We initially trained the model for 80.000 epochs using a patch size of 16x16 pixels and made another test with patch size 40x40 pixels.

It was possible to make the first test using an Nvidia GTX 1080 with 8GB of memory. It took 55 hours to make the full 80.000 iterations, and because of the memory constraint, it was not possible to further increase the patch size in this GPU. We changed configuration using an A100 Tensor Core GPU with 40GB of memory, and it was possible to increase the patch size up to 40x40 pixels, taking 192 hours for the entire train. Due to the lack of this GPU availability, it was not possible to fully train the model and was only possible to complete the first 15.000 epochs, enough for demonstration purposes.

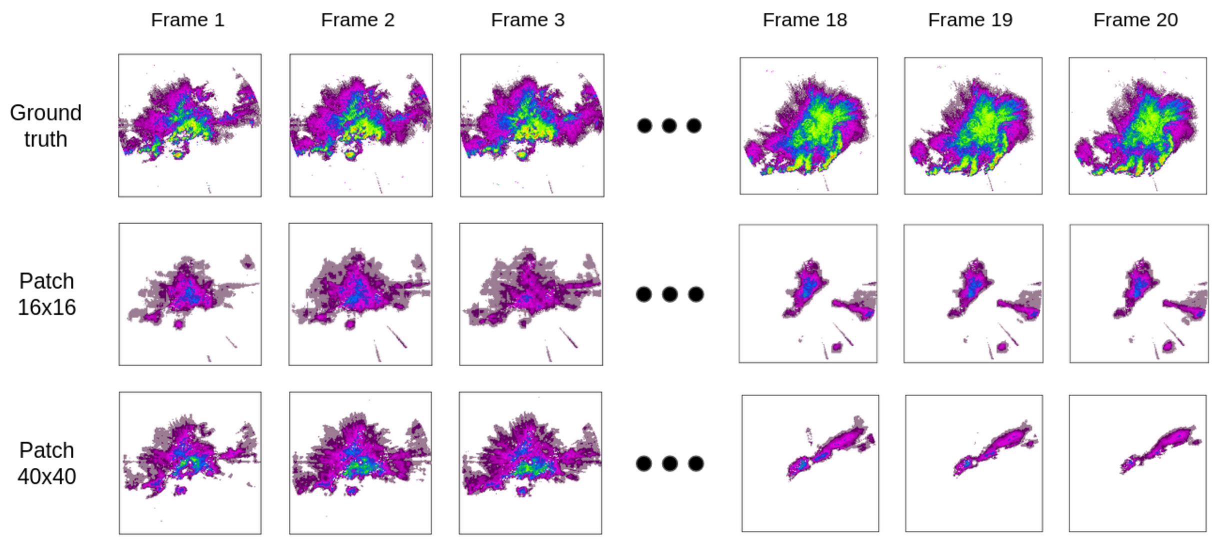


Figure 4.1: IDA-LSTM prediction with patch size 16x16 and 40x40 comparison with ground truth

Results between the two tests are shown in picture 4.1. In both tests, the model was not able to learn the precipitation evolution. In particular, the full train with patch size 16x16 even in the first frames, from 5 to 15 minutes of prediction, was unable to detect high precipitation values, while the model with patch size 40x40 was able to learn high threshold values, while completely ignoring extreme ones. In both cases the prediction in the last frames extremely underestimates the precipitation, missing completely the real evolution.

4.2 Trajectory GRU

The Trajectory GRU model was trained for 200.000 iterations, with 5 input frames, and 20 in output, the former containing the model prediction. The following input thresholds were used to train the model: 0.01, 0.03, 0.1, and 0.3 mm/h. When training the model, the weights were saved every 100.000 iterations and tested across multiple output thresholds as explained in section 3.2.2. Training of the models was done using an Nvidia GTX 1080 with 8GB of memory, taking 96 hours to make the full training and 36 for the test.

These tests were compared using the Critical success index, as explained in 3.2.3, and visualized in the plot 4.2. In this plot, we have the CSI in the y-axis and the output rain threshold in the x-axis. It's important to notice that the x-axis uses a logarithmic scale to represent the rain thresholds.

In this plot, it's possible to see that the performances of the plot tend to degrade slowly for low thresholds, and very quickly for high ones.

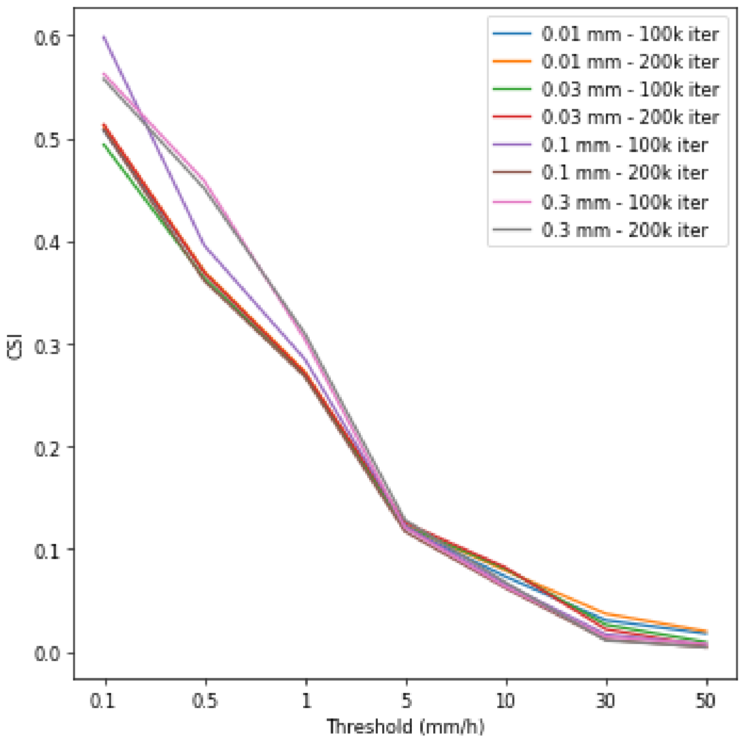


Figure 4.2: Critical success index Trajectory GRU model comparison

In order to get a general idea of the Bias of the model, the rubber plot explained in 3.2.4, was created 4.3. In this plot, it's possible to compare the Critical Success Index using both the input threshold and the output thresholds.

The plot represents lines with different colors for each experiment and symbols for each output threshold. Here it's possible to notice that most experiments tend to overestimate rain lower than the 5 mm/h threshold, while greatly underestimating rainfall with thresholds higher than 30 mm/h per hour.

In these tests, we can see that the various trained models, for high thresholds (e.g. 30 and 50 millimeters per hour) greatly underestimate the amount of precipitation giving relatively few errors, while lower thresholds (e.g. 1 and 5 millimeters per hour) give more false alarms and are more likely to estimate a pixel as rain.

As discussed in section 4.2.4 we chose the model with an input threshold of 0.1 mm/h trained for 100.000 iterations and continued the analysis on that experiment.

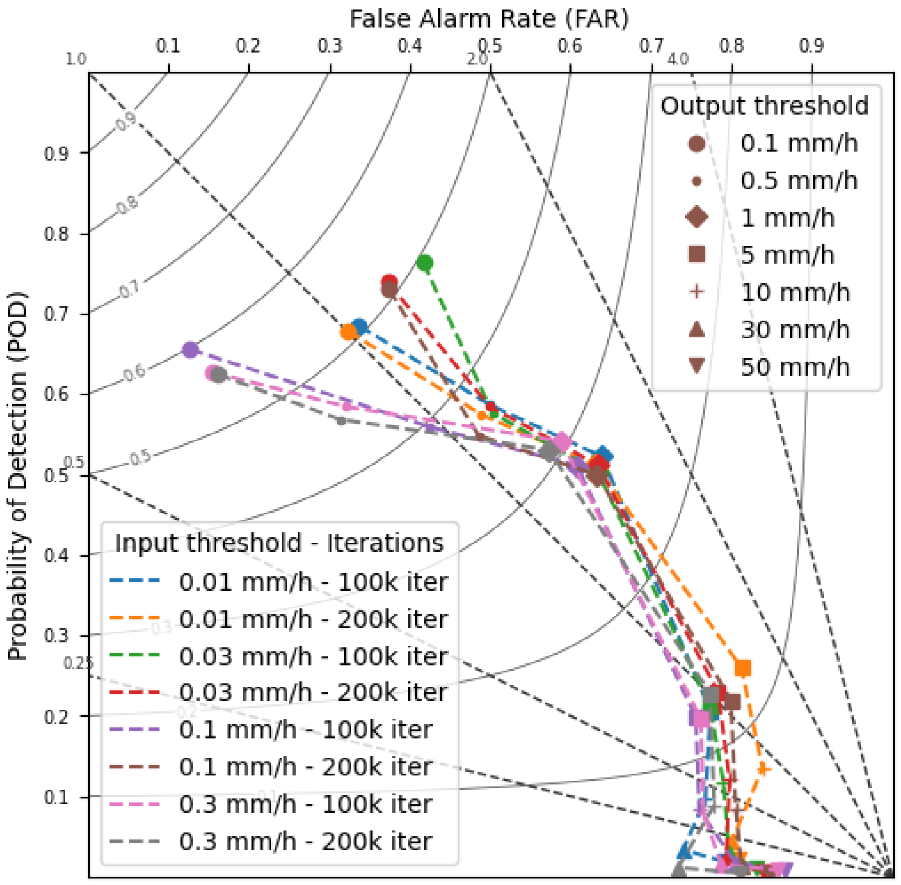


Figure 4.3: Trajectory GRU model comparison with Roebber plot

4.2.1 False alarm rate

In chart 4.4 it's possible to visualize the False Alarm Rate (discussed in 3.2.3) over time, where on the x-axis, we have the lead time of 5 minutes, for a total of 20 output frames, and in the y-axis the false alarm rate.

As you can see from the chart the value of false alarm rate over time increases, since predicting the rain in the correct position becomes more difficult. It's also possible to notice a quick increase in FAR in the first 5 frames for the thresholds values higher than 5 mm/h, opposed to lower threshold values where it's able to give a good estimate for a longer period of time.

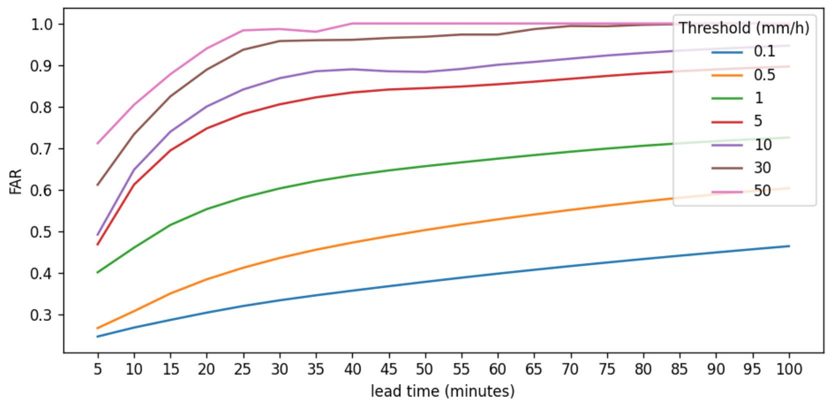


Figure 4.4: Chart Trajectory GRU model performance on False Alarm Rate

4.2.2 Probability of detection

In this chart 4.5, we can see the Probability Of Detection (discussed in 3.2.3) over time. Here it's possible to see that this model is able to make the correct prediction for threshold values over 30 mm/h, only in the first 5 frames while being able to predict precipitation above 10 mm/h for 10 frames.

The probability of detection in this model is very low for low precipitation values, given the well-known problem of vanishing gradient. As explained by Lechner et al. in [15] the Standard RNNs lay in the ordinary differential equation (ODE) representation of the hidden state. For this reason, the usage of this model is limited to applications where the POD is not required for high rainfall values.

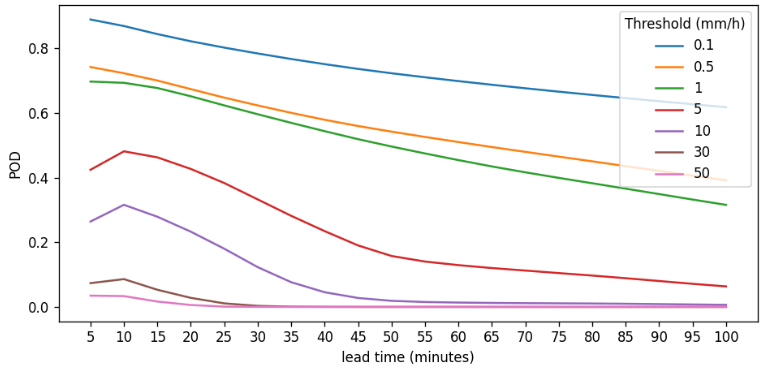


Figure 4.5: Chart Trajectory GRU model performance on Probability Of Detection

4.2.3 Mathematical model

The mathematical model S-PROG was tested amongst the same validation dataset as Trajectory GRU.

In the plot 4.6 it's possible to compare the Critical Success Index over time, of the Trajectory GRU model versus S-PROG. It's possible to notice that the S-PROG model, for thresholds lower than 0.5 mm/h, is able to predict rainfalls in a short period of time, having performance deterioration over the first 10 frames. For precipitation values over 1 mm/h instead is able to predict better rainfall precipitation in the long term, for instance being able to achieve better results after 16 frames for the 5 mm/h threshold value and having always better results for the 50 mm/h threshold.

In the plot 4.7 two models were compared using the Roebber plot. Here it's possible to see that the S-PROG model is able to keep a constant bias on the overall precipitation, while Trajectory GRU strongly underestimates the precipitation for rainfall values over 30 mm/h, but is able to achieve a higher Critical Success Index while keeping a balanced Bias for values under 5 mm/h.

4.2.4 Models comparison

In picture 4.8 it's possible to visualize the comparison between the three models tested and the ground truth on the Moena Flood using the best parameters found.

The results reported in 4.8 and plot 4.1 show that IDA-LSTM was able to learn detailed structures but strongly underestimated prediction of the long term evolution of the precipitation. As discussed in 5.1 this model was discarded in the following comparison, due to the very low performances obtained.

As shown in picture 4.8, Trajectory GRU is able to predict well the low intensity of the precipitation, underestimating the high values, smoothing the extremes, and therefore unable to detect disasters caused by strong rainfall. It was capable of making good long-term predictions, being able to achieve good performances for all thresholds lower than 30 mm/h, as shown in 4.1. For rainfall precipitation above 50 mm/h, the model was unable to achieve good performances, getting at best a critical success index of 0.02.

The S-PROG approach outputs a smooth field and has a lower variance than the original observed field as shown in 4.8. It achieved overall good performances for threshold values over 30 mm/h, having better results than Trajectory GRU for these thresholds. Despite these good results, the prediction comes at the cost of a high False Alarm Rate, leading to the overestimation of precipitation.

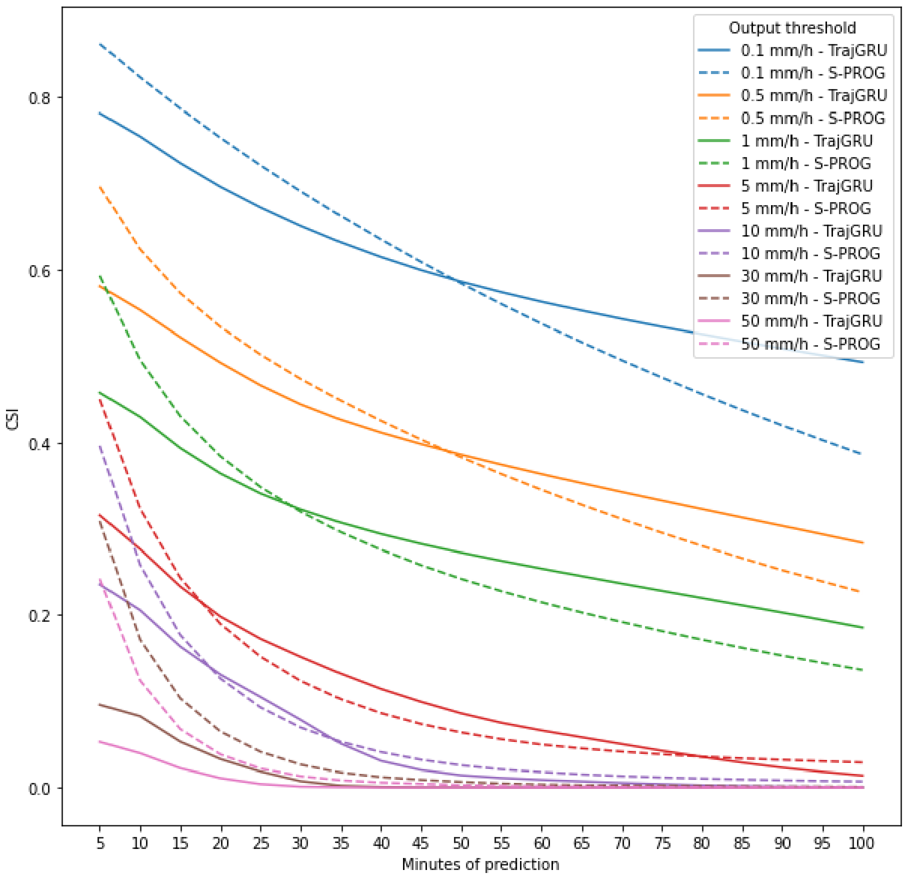


Figure 4.6: Critical Success Index of Trajectory GRU and S-PROG

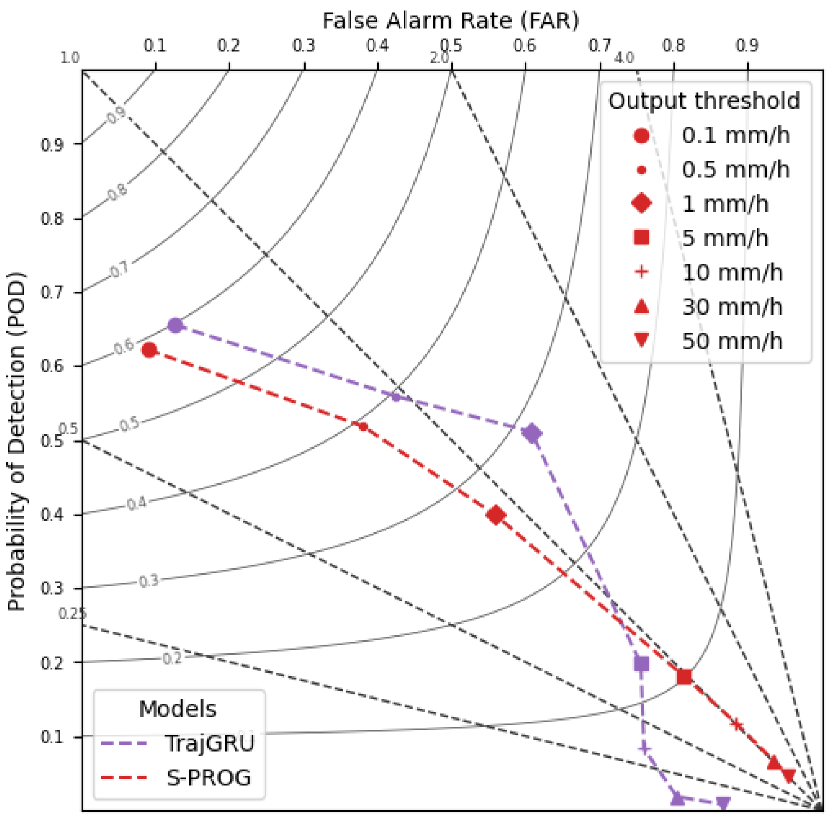


Figure 4.7: S-PROG performance compared with best TrajGRU train, visualized with Roebber plot

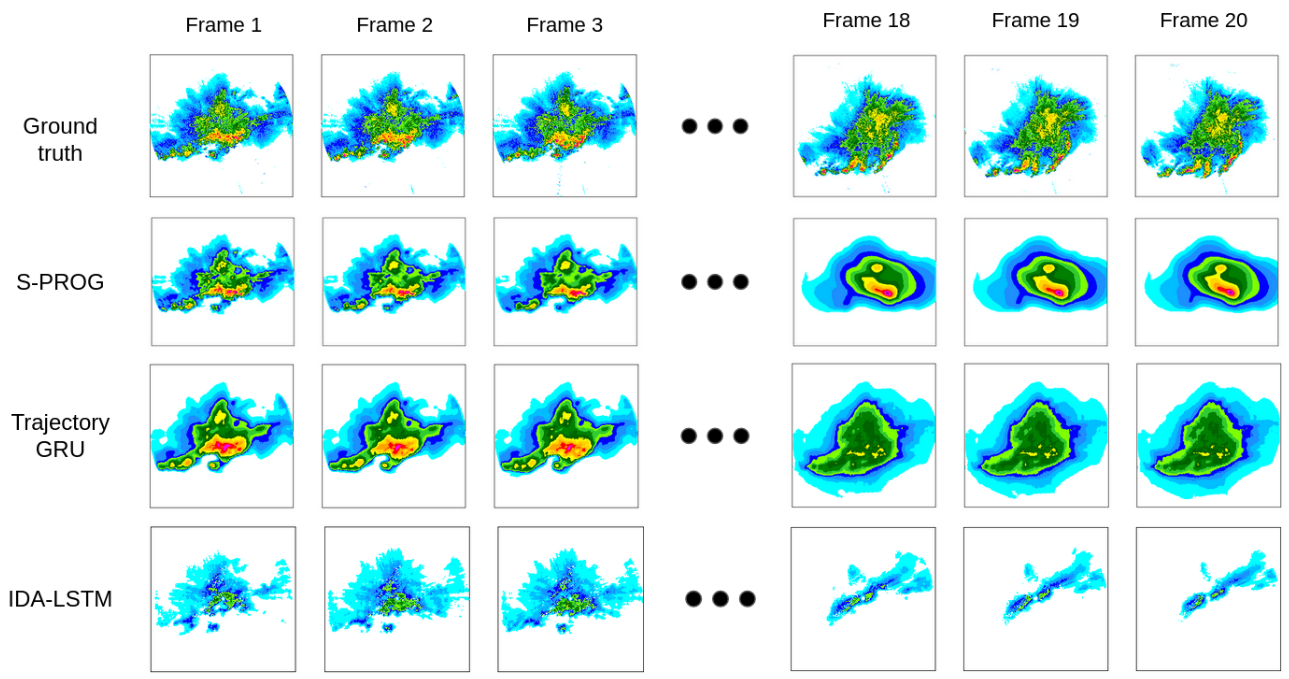


Figure 4.8: Prediction comparison of S-PROG, Trajectory GRU and IDA-LSTM

CSI Threshold (mm/h)		**0.1**	**0.5**	**1**	**5**	**10**	**30**	**50**
S-PROG		0.586	0.393	0.265	0.101	0.062	0.034	**0.023**
TrajGRU 100.000 iters	**0 mm/h**	0.508	0.370	0.271	0.121	0.073	0.031	0.018
	0.07 mm/h	0.494	0.364	0.269	0.123	0.081	0.026	0.010
	0.14 mm/h	**0.598**	0.396	0.285	0.123	0.066	0.017	0.007
	0.28 mm/h	0.562	**0.458**	0.305	0.121	0.063	0.015	0.008
TrajGRU 200.000 iters	**0 mm/h**	0.512	0.370	0.273	0.122	0.079	**0.037**	0.020
	0.07 mm/h	0.513	0.369	0.272	0.126	**0.082**	0.022	0.006
	0.14 mm/h	0.508	0.361	0.268	0.117	0.062	0.012	0.004
	0.28 mm/h	0.557	0.451	**0.310**	**0.128**	0.068	0.011	0.005

Table 4.1: Trajectory GRU and S-PROG model comparison

4.2.5 Datasets comparison

The Trajectory GRU model was trained on both TAASRAD19 and MIARAD datasets.

MIARAD is a dataset that generates the same radar precipitation maps of TAASRAD19, merging the product of two radars. It covers the region of Emilia Romagna, and it outputs the same resolution of 480x480, generating one frame every 5 minutes. The radars cover a wider area, generating images with a spatial resolution per pixel of 1.000 meters.

The model was trained two times per dataset, one on 0.14 mm/h input threshold for rainfall and the other and 0.28 mm/h. Each model was trained for 200.000 iterations, and the weights were saved every 100.000 epochs.

In the results in table 4.2 we can see how the models performed better on low-intensity precipitation for the TAASRAD dataset, while performances in the MIARAD dataset were better for lower thresholds.

Dataset	Iterations	Threshold	0.1	0.5	1	5	10	30	50
TAASRAD	**100.000**	**0.14 mm/h**	**0.598**	0.396	0.285	0.123	0.066	0.017	0.007
		0.28 mm/h	0.562	0.458	0.305	0.121	0.063	0.015	0.008
	200.000	**0.14 mm/h**	0.508	0.361	0.268	0.117	0.062	0.012	0.004
		0.28 mm/h	0.557	0.451	0.310	0.128	0.068	0.011	0.005
MIARAD	**100.000**	**0.14 mm/h**	0.496	0.456	0.416	0.236	0.142	0.041	0.020
		0.28 mm/h	0.522	**0.476**	0.422	0.227	0.133	0.032	0.011
	200.000	**0.14 mm/h**	0.512	0.468	**0.427**	**0.248**	**0.145**	**0.047**	**0.025**
		0.28 mm/h	0.509	0.465	0.418	0.235	0.141	0.040	0.015

Table 4.2: TAASRAD and MIARAD dataset comparison

5 Conclusions

In this work, it was proposed the analysis of the TAASRAD dataset, tested on three different prediction models using multiple parameters to achieve the best results. The dataset was further compared with the MIARAD dataset, using one prediction model, in order to discuss its quality.

In these sections, the methodologies used are discussed underlining limitations, and future works are presented.

5.1 Discussion

Given the results, several remarks about the methods can be made. First, the IDA-LSTM model tested on our dataset was not a reliable baseline for the forecasting abilities of the networks. In all the tests conducted the model was not able to learn the precipitation evolution, strongly underestimating the precipitation from the first frames, and almost completely nullifying the prediction in the last frames.

The motivation behind these poor performances may be related to changes that were done to fit out dataset to the network. The implementation proposed by the authors has the input picture size of 101×101 with a spatial resolution of 3.5 km and a patch size of 32x32, while our implementation had a spatial resolution 7 times smaller. These problems may have been solved by lowering the picture size used in the dataset and using the same spatial resolution, but the project aimed to keep the output of the model the same as the input.

The model S-PROG was trained on the proposed dataset to get a baseline for other models' performance. It uses a mathematical approach to compute the prediction, and it tends to generate a smooth field. The achieved results show a prediction that tends to level every precipitation while overestimating the precipitation in the last frames of the prediction. This was particularly problematic for the aim of the project, indeed this model would output a lot of extreme precipitation warnings and extreme events, leading to taking unhelpful bad weather precautions.

On the other hand, the model Trajectory GRU was tested using multiple input rainfall thresholds and the results were compared against the model S-PROG mathematical approach. The visual results show a model that is able to detect the motion of the precipitation, understanding its trajectory. A problem encountered with the model is that it tends to underestimate the strong precipitation in the long term, leaving little to no extreme prediction. The loss that was supposed to give more importance to the strong precipitation did not seem to give good results for thresholds over 5 mm/h rainfall.

The former model was compared against S-PROG and gave clear signs of the above remarks. Performance of Trajectory GRU was higher for high threshold values and lower for extreme values. This conclusion led us to choose the Trajectory GRU model due to the low frequency of false alarms that the model generates while making a good forecasting prediction.

Fine-tuning of this model was done by comparing multiple input thresholds and testing on different training iteration numbers. There was not a single model standing out from the other, but multiple models had the best results for different thresholds. Given the objective of the model of predicting the extreme weather, we chose the model that had the best results for high thresholds. The one trained for 200.000 iterations and 0 mm/h input threshold had the best results for 30 and 50 mm/h output thresholds, when not considering S-PROG, and the second-best in the 10 mm/h one.

Comparison of the datasets allowed us to compare the quality of the dataset and the results show that MIARAD always exceeded TAASRAD performances for output thresholds higher than 0.5 mm/h. Their differences in performances may be given by multiple motivations, one of them is the spatial resolution of the former being twice the size of TAASRAD. Another motivation may be due to the territory formation of the two regions, as explained in section 1.1.4, that the datasets cover, having Emilia Romagna with 25.1% of mountainous territory, compared to 100% of the region of Trentino [5]. Performance differences are anyway negligible compared to other performance differences given by the different models.

5.2 Future developments

Other models may have been used and tested to get better performance compared to these models. Variation to S-PROG for instance introduces a stochastic approach like ANVIL [1] which adds noise to make the prediction look good where it's not able to predict what is going to happen. As reported by Valentina Gregori et al. this model should achieve better performance compared to S-PROG [12] and these results may have a good performance on high-level thresholds. This method was not considered in this work for the goal of the project but it may be interesting in case false alarm rate would not be an issue for the prediction.

Bibliography

[1] ANVIL nowcast pysteps. URL: https://pysteps.readthedocs.io/en/latest/auto_examples/anvil_nowcast.html#sphx-glr-auto-examples-anvil-nowcast-py.

[2] Average monthly rainy days in trento (trentino alto adige), italy. URL: https://weather-and-climate.com:80/average-monthly-Rainy-days,trento-trentino-alto-adige-it,Italy.

[3] The motion field. URL: https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT12/node3.html.

[4] Performance diagram. URL: https://www.cawcr.gov.au/projects/verification/Roebber/PerformanceDiagram.html.

[5] Principali dimensioni geostatistiche e grado di urbanizzazione del paese. URL: https://www.istat.it/it/archivio/137001.

[6] Radar. Page Version ID: 1075204081. URL: https://en.wikipedia.org/w/index.php?title=Radar&oldid=1075204081.

[7] Vista su bolzano dal monte macaion | alto adige | alto adige, trentino, viaggi e varie | foto racconti di walter donegà. URL: https://foto.walter.bz/foto/alto-adige/penegal-macaion/.

[8] Why is the weather so hard to predict? URL: https://letstalkscience.ca/educational-resources/stem-in-context/why-weather-so-hard-predict.

[9] Chiara Cardinali. Nowcasting - corso radar per operatori dei centri funzionali.

[10] Dan Petersen and Bruce Veenhuis and Greg Carbin and Mark Klein and Mike Bodner Snow. Dan petersen bruce veenhuis greg carbin mark klein mike bodner presentation. URL: https://slideplayer.com/slide/13734399/.

[11] Gabriele Franch, Daniele Nerini, Marta Pendesini, Luca Coviello, Giuseppe Jurman, and Cesare Furlanello. Precipitation nowcasting with orographic enhanced stacked generalization: Improving deep learning predictions on extreme events. 11(3). URL: https://www.mdpi.com/2073-4433/11/3/267, doi:10.3390/atmos11030267.

[12] Valentina Gregori, Ferdinando De Tomasi, Gianluca Ferrari, and Andrea Chini. A comparison of nowcasting methods on the italian radar mosaic. URL: http://master.meteorologiaeoceanografiafisica.unisalento.it/images/students/1920_vgregori/tesi_vgregori_en.pdf.

[13] Charles R. Harris, K. Jarrod Millman, and Stéfan J. van der Walt et al. Array programming with NumPy. 585(7825):357–362. Publisher: Springer Science and Business Media LLC. doi:10.1038/s41586-020-2649-2.

[14] Hilary Weller. Semi-lagrangian extrapolation method. URL: https://www.youtube.com/watch?v=egnsVOvJYIA.

[15] Mathias Lechner and Ramin Hasani. Learning long-term dependencies in irregularly-sampled time series. URL: http://arxiv.org/abs/2006.04418, arXiv:2006.04418.

[16] Chuyao Luo, Xutao Li, Yongliang Wen, Yunming Ye, and Xiaofeng Zhang. A novel LSTM model with interaction dual attention for radar echo extrapolation. 13(2). URL: `https://www.mdpi.com/2072-4292/13/2/164`, `doi:10.3390/rs13020164`.

[17] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61. `doi:10.25080/Majora-92bf1922-00a`.

[18] Aditi Mittal. Understanding RNN and LSTM. URL: `https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e`.

[19] W David Pan, Paul R Harasti, Michael Frost, Qingyun Zhao, John Cook, Timothy Maese, Lee J Wagner, Claude P Hattan, and Bryan T Akagi. A new method for compressing quality-controlled weather radar data by exploiting blankout markers due to thresholding operations. page 6.

[20] Adam Paszke, Sam Gross, and et al. Massa. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc. URL: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[21] S. Pulkkinen, D. Nerini, A. A. Pérez Hortal, C. Velasco-Forero, A. Seed, U. Germann, and L. Foresti. Pysteps: an open-source python library for probabilistic precipitation nowcasting (v1.0). 12(10):4185–4219. URL: `https://gmd.copernicus.org/articles/12/4185/2019/`, `doi:10.5194/gmd-12-4185-2019`.

[22] Suman Ravuri, Karel Lenc, Matthew Willson, Dmitry Kangin, Remi Lam, Piotr Mirowski, Megan Fitzsimons, Maria Athanassiadou, Sheleem Kashem, Sam Madge, Rachel Prudden, Amol Mandhane, Aidan Clark, Andrew Brock, Karen Simonyan, Raia Hadsell, Niall Robinson, Ellen Clancy, Alberto Arribas, and Shakir Mohamed. Skilful precipitation nowcasting using deep generative models of radar. 597(7878):672–677. Publisher: Springer Science and Business Media LLC. `doi:10.1038/s41586-021-03854-z`.

[23] Danny Scrivener. Understanding radar objections for building developments. URL: `https://www.pagerpower.com/news/radar-objection-to-building-development/`.

[24] A. W. Seed. A dynamic and spatial scaling approach to advection forecasting. 42(3):381–388. Publisher: American Meteorological Society. `doi:10.1175/1520-0450(2003)042<0381:adassa>2.0.co;2`.

[25] 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. _eprint: 1706.03458.

[26] Wikipedia. DBZ (meteorology) — wikipedia, the free encyclopedia. URL: `http://en.wikipedia.org/w/index.php?title=DBZ%20(meteorology)&oldid=1012283740`.