COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS

DEEP LEARNING IN ANALYSIS OF PRECIPITATION FROM RADAR DATA

Master's thesis

2019

Bc. Martin Šomodi

COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS

DEEP LEARNING IN ANALYSIS OF PRECIPITATION FROM RADAR DATA

Master's thesis

Study program:	Applied Informatics
Field of Study:	2511 Applied Informatics
Department:	Department of Applied Informatics
Supervisor:	Mgr. Pavel Petrovič, PhD.

Bratislava, 2019 Bc. Martin Šomodi

96885882

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Bc. Martin Šomodi
Study programme:	Applied Computer Science (Single degree study, master II. deg., full time form)
Field of Study:	Applied Informatics
Type of Thesis:	Diploma Thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Deep Learning in Analysis of Precipitation from Radar Data

Annotation: The Slovak Hydrometeorological Institute is a specialized organization providing hydrological and meteorological services at the national and international level. It monitors quantitative and qualitative parameters of the air and water in Slovak territory, collects, verifies, interprets, and archives data and information on the condition and regime of air and water, describes developments in the atmosphere and hydrosphere, issues forecasts, warnings and other information. It makes the data, information and other research available to the public. One of the important tasks is modelling the outflow of the rainfall water. 3D radar data and data from rain gauges are available. The goal of the thesis is to utilize deep neural networks (deep learning) for the analysis of precipitation from radar data in a short time period. The estimated amounts of precipitation in specific time moments and individual localities then enter rain-outflow model of SHMI with the purpose of estimating the localities reached by the rainfall water in the subsequent hours. It is a key task for estimating flood risks and publishing warnings on excessive rainfall.

Literature: Aaron Sim: Estimating Rainfall From Weather Radar Readings Using Recurrent Neural Networks, Theoretical Systems Biology Laboratory at Imperial College London, UK, December 09, 2015.

Keywords: deep learning, meteorology, radar data, precipitation

Supervisor:	Mgr. Pavel Petrovič, PhD.
Consultant:	Mgr. Jozef Vivoda
Department:	FMFI.KAI - Department of Applied Informatics
Head of department:	prof. Ing. Igor Farkaš, Dr.
Assigned:	21.04.2018
Approved:	23.04.2018

prof. RNDr. Roman Ďurikovič, PhD.
Guarantor of Study Programme

....

Student Supervisor

I hereby declare that I wrote this thesis by myself, only with the help of the referenced literature and thesis consultant, under the supervision of my thesis advisor.

. . . .

Bratislava, 2019 Bc. Martin Šomodi

Acknowledgement

I would like to thank to my advisor Mgr. Pavel Petrovič, PhD. for enabling me the opportunity to work on this assignment and his guidance throughout my work on this thesis.
I would also like to express my gratitude to my family and close ones who gave me support and means to complete this thesis.
Special thanks goes to Mgr. Jozef Vivoda from the Slovak hydrometeorological institute, who guided me through hydrometeorology details with his expertise.

Abstrakt

V práci prezentujeme umelé neurónové siete vhodné na analýzu zrážok. Natrénovali sme viacero modelov neuronových sietí na radarových dátach od Slovenského hydrometeorologického ústavu.

Vytvorili sme hlboké obojsmerné rekurentné modely neurónových sietí, ktoré 8-násobne prekonali referenčné hodnoty poskytnuté Slovenským hydrometeorologickým ústavom. Modely vytvorené v tejto práci pomôžu Slovenskému hydrometeorologickému ústavu s analýzou zrážok pre účely dodatočného skúmania a predpovede povodní.

Kľúčové slová: deep learning, meteorológia, radarové údaje, zrážky

Abstract

In the work we present artificial neural networks suited for precipitation analysis. Multiple neural networks are fitted on radar data provided by Slovak hydrometeorological institute.

We created deep bidirectional recurrent neural networks, which achieved approximately 8 times better results than reference values provided by Slovak hydrometeorological institute. Models created for the purpose of this work shall help Slovak hydrometeorological institute with rainfall analysis for additional calculations and flood prediction purposes.

Keywords: deep learning, meteorology, radar data, precipitation

Contents

Introduction

Artificial intelligence helps, improves, takes over various scientific branches. However, there are fields where methods of AI expanded to level of utmost importance, yet there are spheres with very little touch of AI.

Hydro-meteorology is a discipline where various machine learning methods are regularly used, though some of the predictions and analysis still depend on old very deterministic-like calculations. We believe that many of those problems can be regulated or even completely solved using modern machine learning techniques.

Rainfall analysis is a task, that is generally approached in an old-fashioned way in Europe. The Slovak hydrometeorological institute uses radar reflectivity data to calculate precipitation using Marshall-Palmer distribution. This method tends to be rather inaccurate, due to mostly due to high measurement error and significant noise in radar measurements.

The main goal of the thesis is to create an artificial neural network capable of predicting in-progress rainfall using data from Slovak radar network. Neural network predictions provide data for further analysis, most importantly estimating flood dangers in certain regions of Slovak Republic.

The first chapter is an overview of theoretical background of the methodology used in thesis with focus on artificial neural networks and rainfall analysis. The second chapter introduces data and describes means of utilizing the neural network model. The third chapter provides results of the final model, the comparison of different approaches and a performance evaluation in general.

Chapter 1

Overview

This chapter is the general overview of theoretical knowledge and historical background of methods used in the thesis. The first two sections focus on neural networks, while the third section brings details about rainfall analysis in general and in Slovak Republic specifically.

1.1 Artificial Neural Network

"In its most general form, a neural network is a machine that is designed to model the way in which the brain performs a particular task or function of interest." [Hay09]. Artificial neural networks were created to reflect the model of human brain. Haykin expressed the resemblance of a brain and an artificial neural network in two important points:

- Knowledge is obtained from the network's environment through the course of learning.
- Large degree of interconnection is used to accumulate the acquired knowledge.

As neural networks being currently the most powerful tool of modern artificial intelligence, they have been through lot of evolution and experimentation.

1.1.1 Early history of Neural networks

The first model of neural network was created in 1943 by McCulloch and Pitts [MP43]. Threshold Logic Unit, as the first artificial neuron of the model, was capable of simulating elementary logical operations (such as AND, OR, NOT). This type of neuron was not capable of learning whatsoever and threshold values were anchored. Figure 1.1 displays McCulloch's and Pitts' formal neurons.

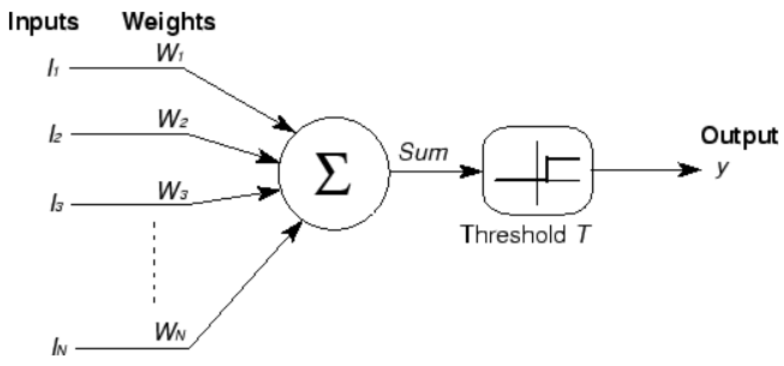


Figure 1.1: Threshold Logic Unit designed by McCulloch and Pitts.

Frank Rosenblatt later modified McCulloch-Pitts model, creating an algorithm he called perceptron, using which, the model was able to do the task of pattern recognition. The image of Rosenblatt's perceptron is shown in figure 1.2. However, most of the progress in the area was dampened by the research by Minsky and Papert in 1969. They pointed out some of the limitations of perceptron. The main problem was that the perceptron is unable to solve linearly inseparable problems (XOR problem). Scientists were yet unaware of any learning rules mandatory for resolving such tasks, therefore Minsky and Papert suggested focusing on different approaches [MP69]. The research for supervised learning in neural networks was delayed and later this branch was reborn with the invention of error back-propagation learning [KBP+97].

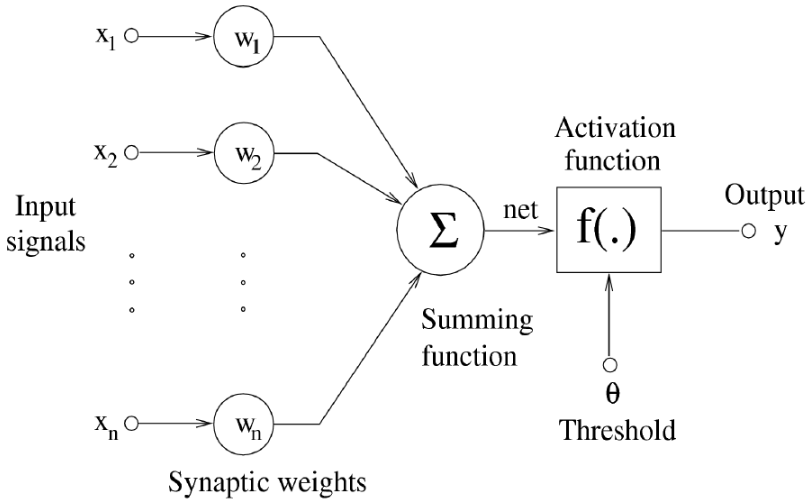


Figure 1.2: Rosenblatt's perceptron.

1.1.2 Multi-layer perceptron and back-propagation

Back-propagation was first practical method used for neural network models with hidden layers. The learning rule is based on minimizing the error function, which defines the difference between desired and actual output of the network. With back-propagation available, neural networks with hidden layers showed a great promise in many fields of science. In addition to this, multi-layer perceptron is able to approximate arbitrary functions, making them a very valuable tool for regression [KBP+97]. The architecture of MLP with one hidden layer is shown in figure 1.3.

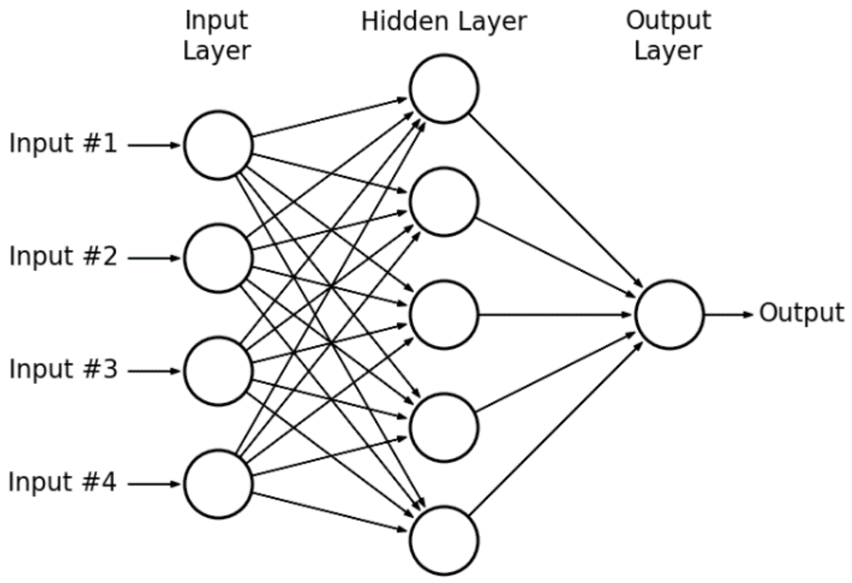


Figure 1.3: Multi-layer perceptron model.

In spite of considerable success in form of back-propagation algorithm, neural networks were commonly replaced by different machine learning methods in 1990's. The paper in 1992 brought up more attention to support vector machine model [BGV92]. Since SVM models were easier to apply, fitting was faster and choice of hyper-parameters was simpler, neural networks lost on popularity again [Kuz14].

1.1.3 Deep learning

Nowadays *deep learning* is commonly used as a phrase in connection with artificial intelligence, however it was only in late 2000's when the real progress in this area rose. Very significant research in 1991 by Sepp Hochreiter revealed a considerable problem in using a gradient descent algorithm for back-propagation in networks with more hidden layers [Kuz14]. This obstacle, called the problem of *vanishing/exploding* gradients, consists in error passing back through layers is exponentially augmented in each layer. As a result of this, the gradients in initial layers tend to be extremely large or small. This problem is usually solved by pre-training the network with unsupervised learning or different choice in activation functions. Avoiding activation functions which have domain containing potentially very large numbers, while range consists of non-proportionally small numbers helps. The simple example of this is a logistic sigmoid function,

$$f(x) = \tfrac{1}{1+e^{-x}},$$

which have domain of all real numbers, while range is $<$0,1$>$. An activation function that suffers less from the problem of vanishing/exploding gradients is a rectified linear unit,

$$f(x) = max(0, x),$$

since the derivation of the positive part is constantly 1. The derivation of non-positive part is 0,which causes sparsity in gradients and activations, therefore speeds up the training [Kuz14].

The plot showing a comparison of logistic sigmoid and rectified linear unit functions is displayed in the figure 1.4

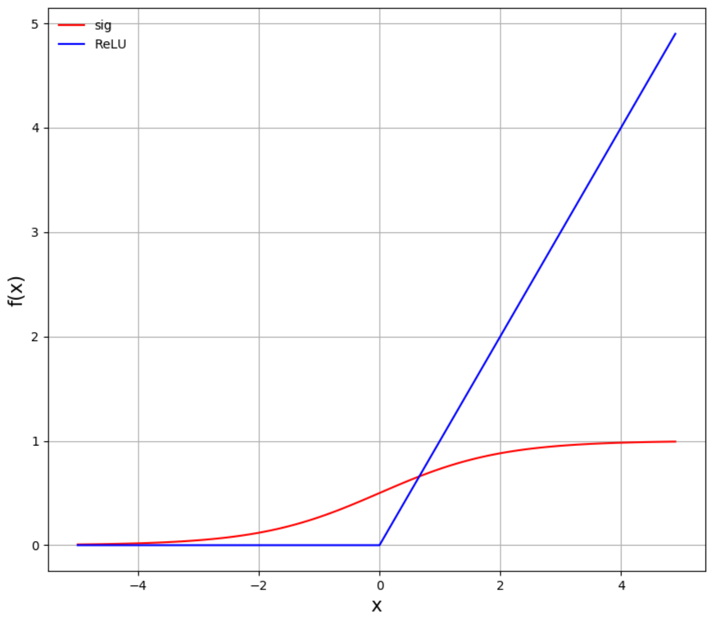


Figure 1.4: The comparison of logistic sigmoid and ReLU for x in range <-5, 5>

Deep neural networks got more popular lately, because of the graduate increase in computing power. Since deep networks are suited for molding complex, nonlinear transformations, various model architectures achieve remarkably well in specific tasks.

1.2 Recurrent Neural Network

Although simple deep architectures are capable of solving various tasks, with some category of tasks these networks still seem to have troubles. Particularly problems, where the essence of the task itself is quite unconventional, like tasks with time dependency [KBP+97]. Let's assume following sequence of vector pairs (x,y):

$$a \to X; b \to Y; c \to Z; d \to W,$$

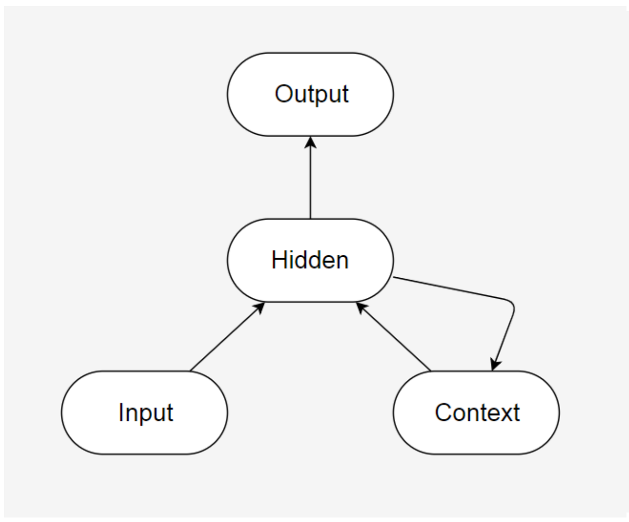
where $a, b, c, d \in R^n$ and $X, Y, Z, W \in R^m$. Elements of this sequence form a certain structure in space $R^n \times R^m$ and this structure can be simply fitted by MLP model. However if we alter the sequence in a specific way:

$$a \to X; b \to Y; c \to Z; b \to W; a \to X; b \to Y,$$

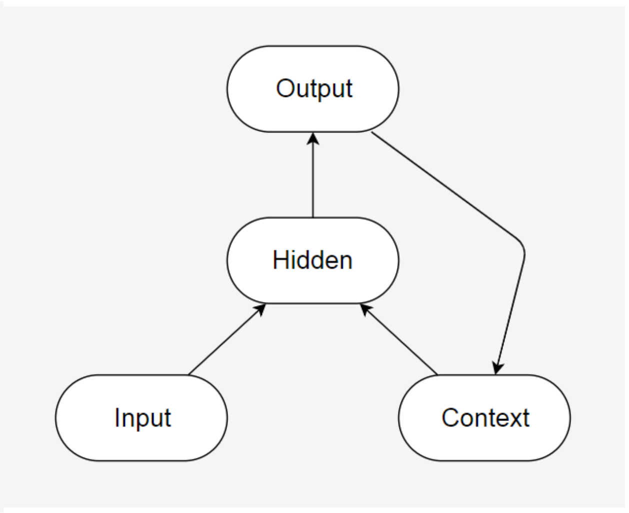
we can observe a very special pattern with the significance to b. In the sequence, b outputs differently according to its predecessor in the sequence. Therefore, if we wanted to fit the model, the output of the network should not be influenced only by its input. The information about the past inputs is supposed to be stored in the network as well.

Generally, any artificial neural network can be classified as recurrent, if the network is capable of storing the information about neuron activation from previous inputs. This is usually done by dedicating some of the neurons of the network for storing this kind of information.

Simple architectures, like Elman's network [Elm90] and Jordan's network [Jor89] are elegant extensions of MLP (covered in section 1.1.2), with added context layer adding loop into network. Elman's hidden layer outputs both into output layer and context layer. The input for hidden layer is input layer and context layer.



(a) Elman's network



(b) Jordan's network

Figure 1.5: The difference between Eman's and Jordan's network.

Difference in Jordan's network is that context layer's input comes from network's output layer. Figure 1.5 shows the architectures of Elman's network and Jordan's network.

Bengio proposed having multiple context layers, as multiple layer's activation memory [KBP+97]. The architecture created as a combination of Jordan's and Elman's network is displayed in figure 1.6.

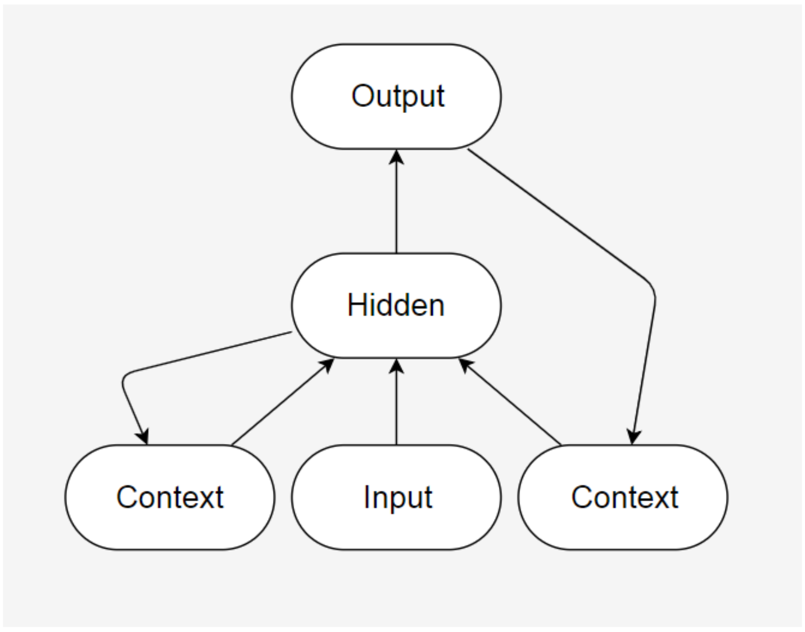


Figure 1.6: Combination of Jordan's and Elman's network.

Recurrent neural networks got highly popular for time-dependent tasks, sequence prediction or generation, speech recognition, handwriting recognition, etc. As a result of this, large numbers of various RNN architectures were created. With each task, different architectures showed promises. This gave rise to new learning methods used in recurrent neural networks [KBP+97]. As a modification of BP algorithm (see 1.1.2), back-propagation through time was created. In BPTT recurrent network is unfolded into multi-layer perceptron. Then usual back-propagation algorithm is used to update the weights of unfolded network. Another widely used algorithm created for this purpose is called real time recurrent learning. The changes in network weights are accomplished while the network continues to perform [Hay09].

Both methods are based on gradient descent and therefore are inclined to be involved in problem of vanishing/exploding gradients described in section 1.1.3.

1.2.1 Similar work

The success of using recurrent neural networks for rainfall analysis was achieved by Aaron Sim in 2015. He attended a Kaggle-hosted competition named *How Much Did It Rain? II*, which he won using recurrent neural networks. The goal of the competition was to predict hourly rainfall from radar measurement sequences. Aaron's article *Estimating rainfall from weather radar readings using recurrent neural networks* explains his approach to this problem and reasons why he achieved such notable success in the competition [Sim15].

Architectures created for the purpose of the competition inspired networks in the thesis. However, an approach to data handling was completely different, due to differences in data.

1.3 Analysis of rainfall

Analysis of rainfall is a significant segment in hydrometeorology. It connects to modelling the outflow of the rainfall water and then later estimating flood dangers.

Data provided by Slovak Hydrometeorological Institute are gathered using over 120 precipitation gauges and radar measurements. Precipitation gauge measurements are used as target values, or *ground truth.* Meteorologic radar is a device, capable of detecting intensity of precipitation. Radar emits cone-shaped pulses into atmosphere, where the energy collides with any objects - drops of water, snowflakes, partly clouds, terrain, airplanes, etc. Portion of energy echoes back from targets and is detected by radar. Distance r can be then calculated by formula

$$r = \tfrac{tc}{2},$$

where t is the time between the pulse being dispatched and the energy being received and c is the light speed. The wavelengths used by radars filter meteorologic consisting highly of water and ice particles, because they are within the range of Rayleigh scattering. Radar reflectivity Z then varies by the sixth power of the rain droplets' diameter D:

$$Z = \sum(D^6),$$

in decibel, being logarithmic of its original mm^6/m^3, for practical reasons. This measurement occurs under multiple angles, therefore reflectivity is stored for multiple altitudes [Bli11].

1.3.1 Slovak radar network

Slovak Republic disposes of 4 weather radars strategically positioned over the republic. Previous network consisting only of two radars: Maly Javornik (west part of Slovakia) and Kojsovska hola (east part of Slovakia) was expanded in 2015, because the network appeared to be insufficient. The exact location of radars is displayed in figure 1.7.

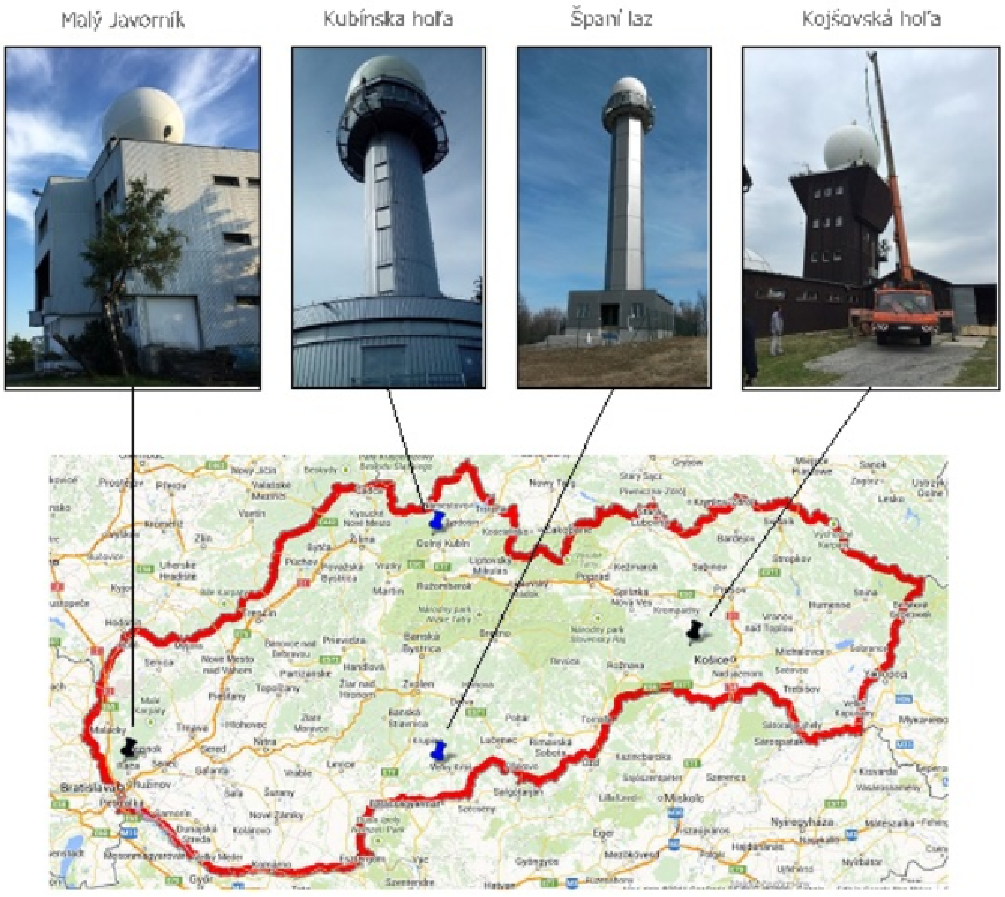


Figure 1.7: Slovak radars placement and appearance.

With the addition of radar stations: Kubinska hola (north part of Slovakia) and Spani laz (central south Slovakia), the network coverage was improved [JKOM14]. Current radar network and its coverage is illustrated in figure 1.8.

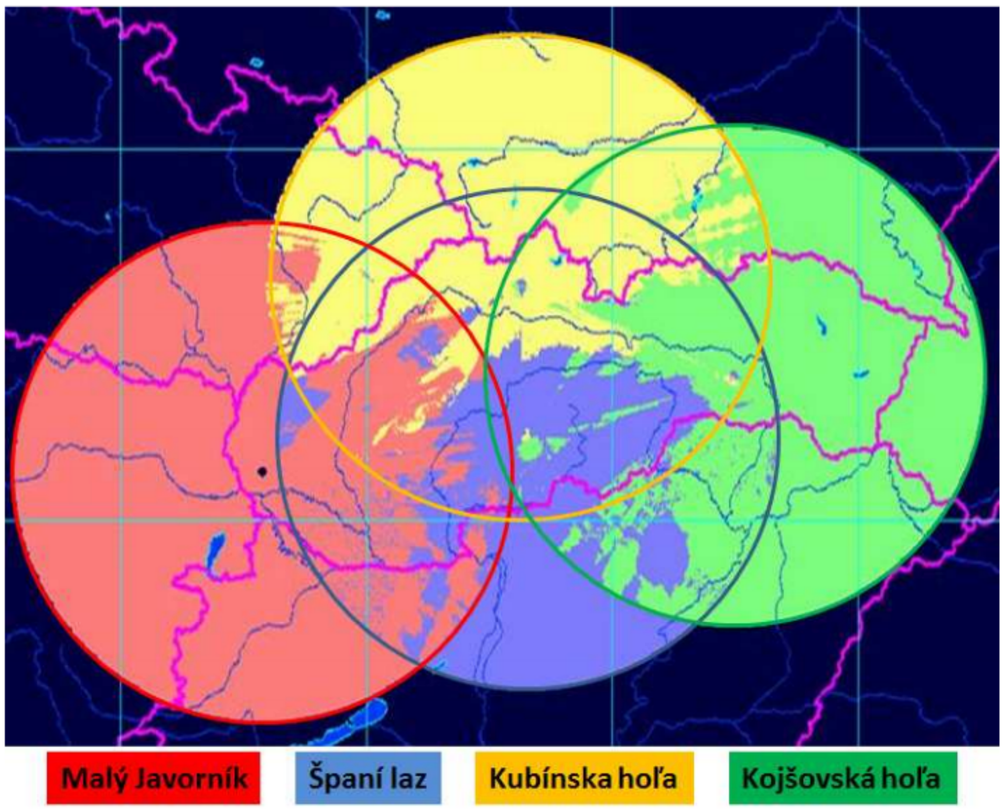


Figure 1.8: Map of Slovak radar network with each radar's range.

Chapter 2

Problem Formulation and Analysis

The Problem Formulation and Analysis chapter introduces some data analysis, approach to data preprocessing and programming languages and tools used in work.

2.1 Dataset

The Slovak Hydrometeorological Institute provided us with precipitation gauges measurements and radar data. Dataset was collected over the course of 5 months, August through December 2016, with 5 minute measurement intervals.

2.1.1 Data analysis

The complete dataset consists of following parameters:

- id of station

- country
- radar name/code
- radar measurement
- altitude of radar measurement
- calculated rainfall using Marshall-Palmer distribution
- longitude
- latitude
- time-stamp
- quality flag
- precipitation measurement

During the measurement, a radar gained up to 10 reflections from different altitudes. Therefore the data for a 5-minute window contains 10 radar measurements, altitudes and rainfall values calculated using Marshall-Palmer distribution. The table 2.1 displays some first-look data properties. Parameters *altitude[i]*, *reflectivity[i]* and *marshall-palmer[i]* corresponds to altitude of a measurement, radar measurement in dBZ and rainfall calculated using Marshall-Palmer distribution. Parameters x and y are values obtained by doing a projection from longitude and latitude and dividing by a common constant due to their initial disproportional scale. *Target rainfall* refers to precipitation measurement in mm/5 min, being the value we wish to approximate.

Table 2.1: Input data properties

Parameter name	**Mean**	**Standard deviation**	**Max value**	**Min value**
altitude0	2550.71	1303.096	6302.123	257.657
reflectivity0	-29.766	9.847	63.013	-32.
marshall-palmer0	0.035	0.548	316.369	0.
altitude1	4026.842	2141.568	10231.612	661.502
reflectivity1	-30.079	9.289	60.12	-32.
marshall-palmer1	0.032	0.508	208.622	0.
altitude2	5611.767	3098.719	14413.533	0
reflectivity2	-30.445	8.279	61.347	-32.
marshall-palmer2	0.023	0.422	248.925	0.
altitude3	7288.891	4099.346	18750.946	0.
reflectivity3	-30.745	7.368	62.26	-32.
marshall-palmer3	0.016	0.363	283.86	0.
altitude4	9675.286	6185.748	27906.519	0.
reflectivity4	-31.01	6.487	61.628	-32.
marshall-palmer4	0.011	0.306	259.171	0.
altitude5	12787.393	8874.292	37419.656	0.
reflectivity5	-31.138	5.862	61.106	-32.
marshall-palmer5	0.008	0.258	240.443	0.
altitude6	16604.778	12296.689	49720.138	0.
reflectivity6	-31.235	5.386	64.24	-32.
marshall-palmer6	0.005	0.213	377.444	0.
altitude7	19112.118	17773.987	65532.615	0.
reflectivity7	-27.61	11.159	58.257	-32.
marshall-palmer7	0.003	0.165	159.553	0.

altitude8	22230.871	24440.53	83446.11	0.
reflectivity8	-23.057	14.408	59.835	-32.
marshall-palmer8	0.002	0.104	200.24	0.
altitude9	23339.289	32487.041	103938.695	0.
reflectivity9	-16.127	16.014	58.066	-32.
marshall-palmer9	0.001	0.065	155.234	0.
x*	-0.397	0.097	0.	-0.568
y*	-1.233	0.039	0.	-1.327
target rainfall	0.074	0.95	99.12	0.

2.1.2 Data preprocessing

The process of data breakdown and analysis revealed that preprocessing is the essential process for subsequent stages of model creation.

The first round of data preprocessing simply joined data rows, since obtained data were in multiple files. Filtering out samples with insufficient *quality flag* ensured the removal of transparently incorrect samples.

Since we use recurrent models of neural networks (specifics in 3.1), it is crucial that the data entering the network are in certain order. Seeing that, the dataset needed to be sorted according particular rules. First thing was categorizing data by *station id.* In total, we split 19,977,638 samples into 137 stations, averaging 145,822 samples per station. Each station data was then sorted by *time-stamp* and replicated to create hour periods. Each hour period consists of 12 time frames. The figure 2.1 shows the process of duplicating sorted samples to create final samples, which contain information from complete hour.

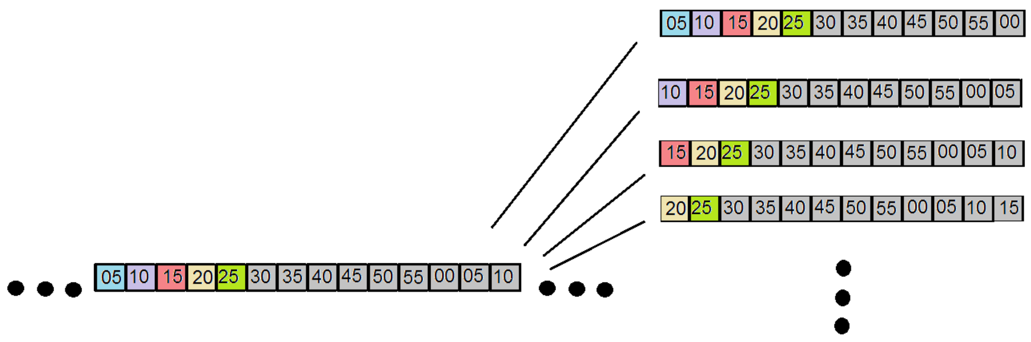


Figure 2.1: The data duplication into 12-frame windows.

At this point the first attempts for model fitting were made. Nevertheless, the model overwhelmed by vast majority of "empty" data samples* was not able to predict any reasonable values (*data where there was no rainfall, nor radar measurements at all). The model prediction average error was incredibly small for one simple reason. The model simply predicted almost 0 rainfall in every sample, which meant it would be correct in over 99% cases. However, this result is unacceptable for our purpose of using predictions to forecast flood dangers. In our case, it is far worse to ignore a massive rainfall than to predict one, when there is none. Because of this we closely analyzed distribution of rainfall across the dataset. The figures 2.2 and 2.3 show the distribution of data among 9 range groups.

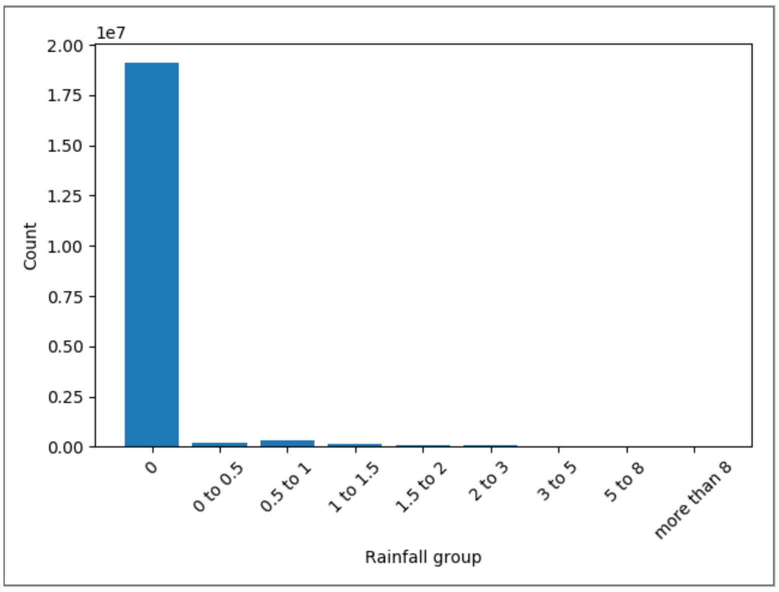


Figure 2.2: Precipitation distribution across the dataset.

The figure 2.2 clearly shows that samples with 0 mm per 5 minutes are very frequently represented in our dataset. With this in mind, we can better understand why model is ignoring samples with more rainfall, especially those with more than 8 mm per 5 minutes. If we ignore the first group, the distribution of other groups becomes more balanced, which is shown in figure 2.3.

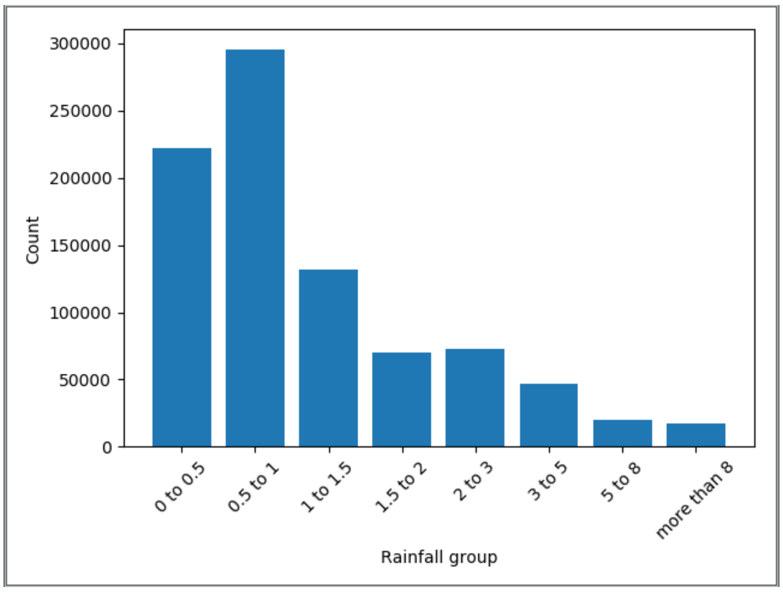


Figure 2.3: Precipitation distribution across the dataset without 0 mm/5 min samples.

In order to improve the model performance in situations with major precipitation volume, the input data needed to be normalized. We simply ignored the majority of empty samples and filtered samples to create custom samples ratio. We started out at the uniform distribution and through experiments decreased the ratio of groups with higher rainfall level. The best created distribution ratio is shown in table 2.2.

Rainfall group	Group ratio
0	10
0 to 0.5	10
0.5 to 1	10
1 to 1.5	7.5
1.5 to 2	5
2 to 3	5
3 to 5	5
5 to 8	5
more than 8	5

Table 2.2: The best ratio of rainfall groups for input.

The procedure of data distribution normalization showed the first reasonable results in model fitting. At this point, it was obvious that the models are beginning to approximate our targets for the first time.

However, there was another substantial obstacle models were facing and that being noise in radar measurements. Upon closer inspection of data, there still was a large number of misleading samples - ones that contain no reflectivity in radar measurements, but notable precipitation levels. Samples like that are caused by radar *measurement uncertainty*. The figure 2.4 plainly displays some of the possible causes of measurement uncertainty. By further filtering such data samples out, we created a final set of viable input samples for neural network.

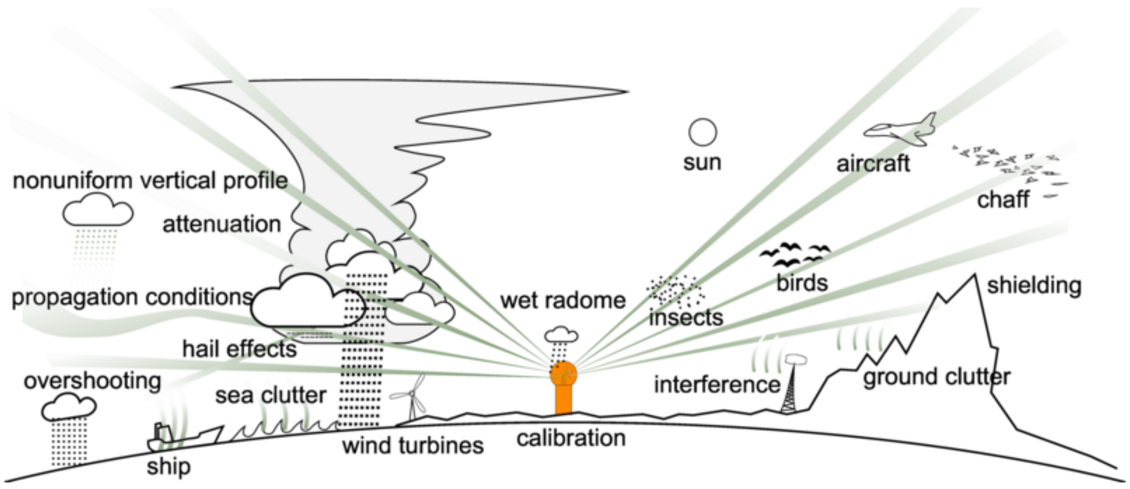


Figure 2.4: Common sources of anomalies in radar measurements

Chapter 3

Solution Design

The third chapter describes software used and neural network models utilized for the thesis.

The process of creating an architecture for neural network models was highly inspired by Aaron Sim's *kaggle* competition winning article [Sim15]. Since our task is a regression based on data from time series, we deploy recurrent models (reasons for RNN are explained in 1.2).

3.1 Implementation details

All of the model fitting, data analysis and preprocessing were written in python 3.6, with use of some libraries commonly used for data science and neural networks.

More and more popular *Numpy* library was used for most of the data handling and any work with matrices and tensors. Neural network models were created using *keras* with GPU-accelerated *tensorflow* backend facilitated through NVIDIA's CUDA platform. And most of the data visualization and results evaluation were produced using *matplotlib*.

3.2 Evolution of models

We started out with a simple many-to-one recurrent neural network. The figure 3.1 displays this simple, shallow model.

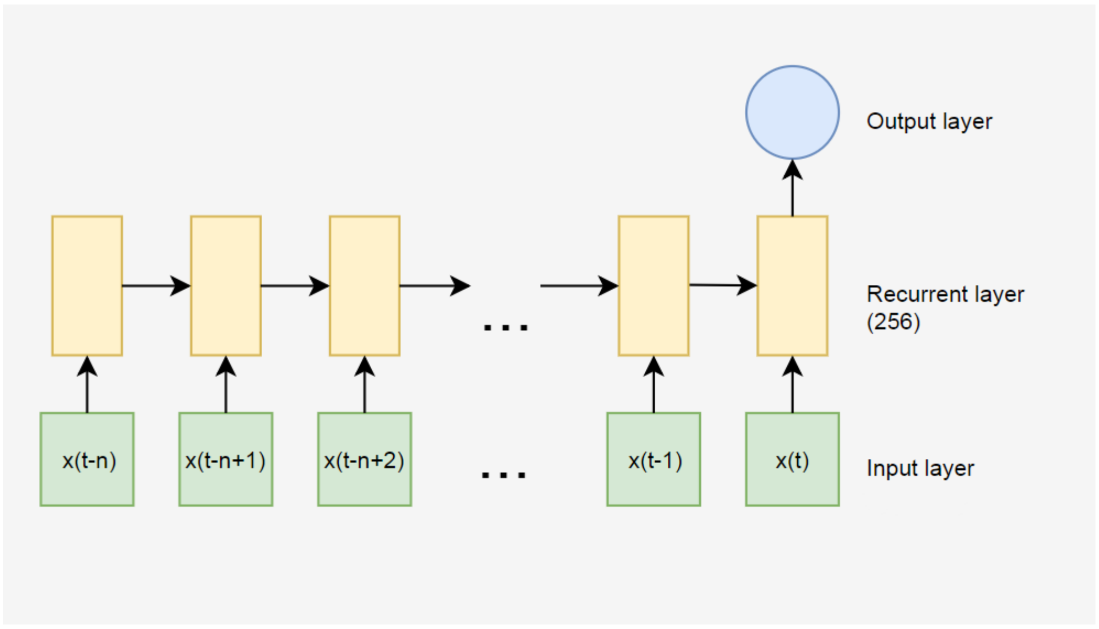


Figure 3.1: First RNN model, consisting of input, single recurrent and output layers.

Inspired by the Aaron Sim's article, we implemented bidirectional recurrent layer, as we can consider the task in a different matter. Aaron wrote "there is nothing preventing us from viewing the problem as rain flying up from rain gauges on the ground and reconstituting itself as clouds" [Sim15] and introduced a reversed recurrent layer on the top of the original one. Results (more in detail in chapter 4) show that the architecture with added reversed recurrent layer surpassed the simpler model. The model with bidirectional recurrent layer is shown in figure 3.2.

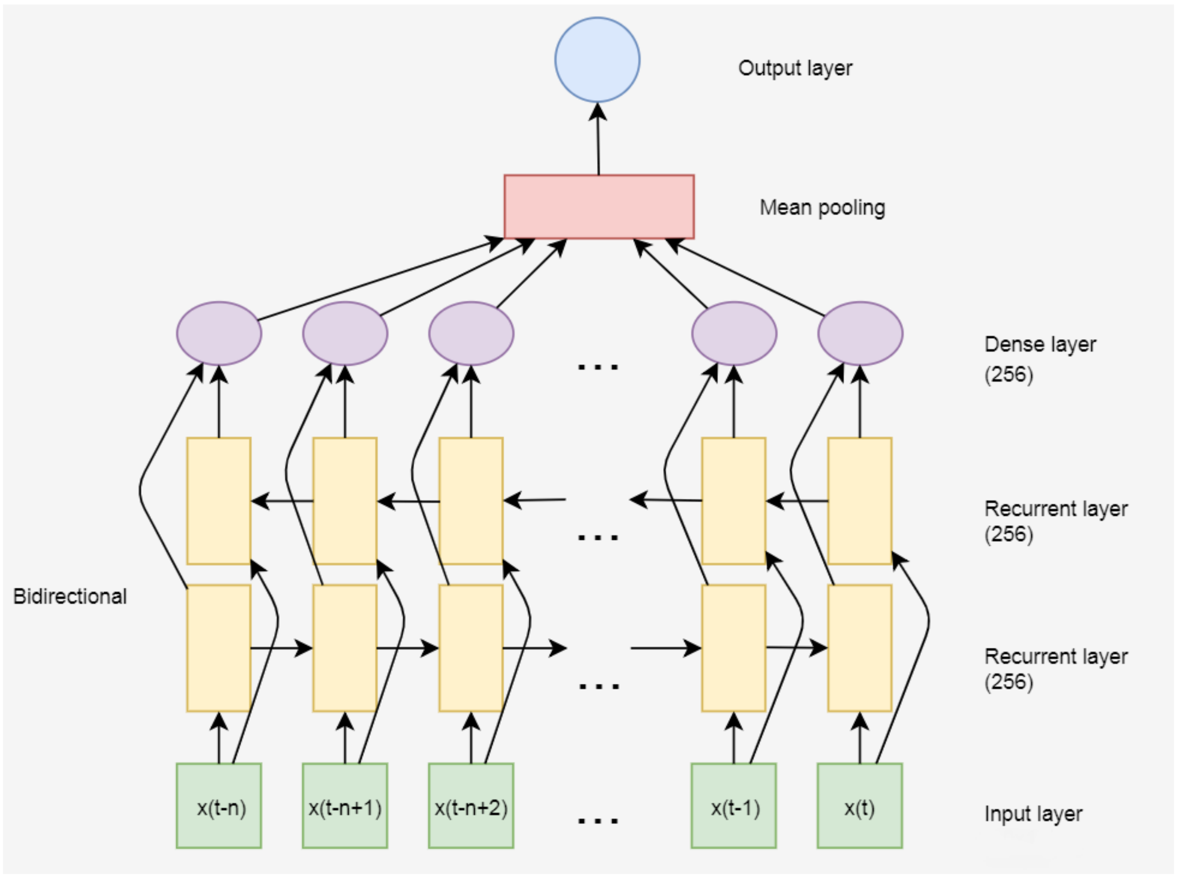


Figure 3.2: RNN model, containing bidirectional recurrent layer.

Model was extended by another stack of bidirectional recurrent layer gaining more depth. Neurons count rises towards the output layer through the network. A set of experiments was executed to test the best neuron counts performance. The performance peaked with the combination of 256 neurons in the first layers and 512 neurons in deeper layers of the network. The figure 3.3 captures the final architecture of model used for fitting and further experiments described in the next section (sec. 3.3).

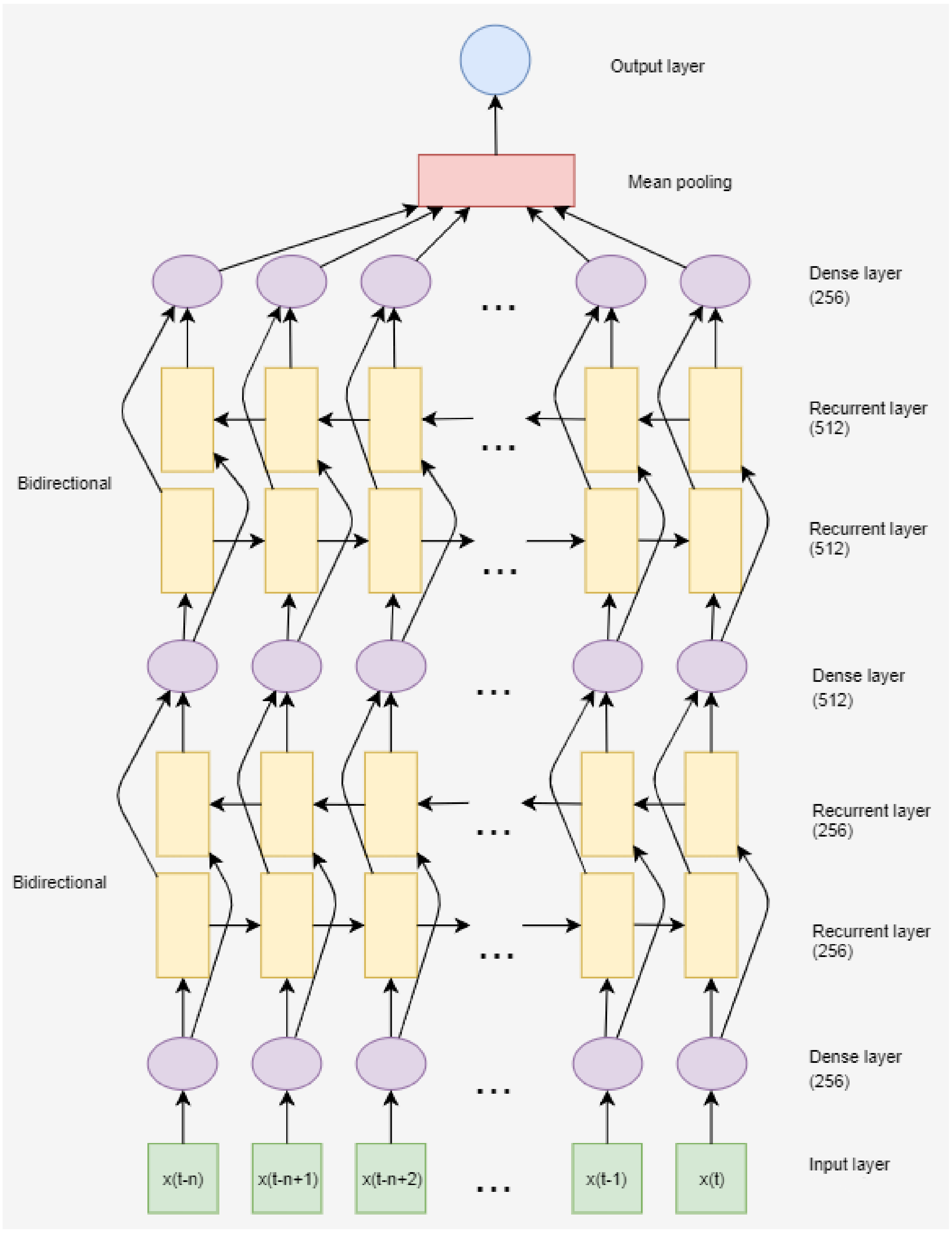


Figure 3.3: Final RNN model with best overall performance.

3.3 Multiple models

To further improve performance, we decided to try an experiment where isolated model for each station was created. The final result contains 137 models for all stations and one general model trained on data from all stations. The results are compared in the chapter 4, Results and Evaluation.

Chapter 4

Results and Evaluation

This chapter focuses on evaluation of results. The first section explains the interpretation of results and graphs, and other sections provide the results from different points of view.

4.1 Means of performance evaluation

The ground truth in the context of the evaluating the results are values used as target for neural network measured by precipitation gauges, described in section 1.3. The results evaluated in this chapter are judged by following assessments:

- mean squared error (MSE)
- predicted value vs ground truth graphs (2 types)
- comparison of predicted and reference values

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2,$$

where Y is predicted value, $\hat{Y}$ is expected value and n is the number of samples.

The figure 4.1 is an example of "predicted value vs ground truth graph". Blue dots represent the relationship between the predicted value and the target. In ideal situation, all of the blue dots should be on the orange line. A dot located closer to x axis means the prediction was smaller than it should have been. On the other side, if a dot is leaning towards y axis, the prediction value was higher than actual value.

The second type of "predicted value vs ground truth graph" is presented by example on the figure 4.2. The y axis represents rainfall value, while x axis is only sample rank. Samples were sorted in ascending order by target values for clearer comparison. In ideal situation, blue dots should all be covered by orange dots, meaning all of the samples were predicted exactly the same as the ground truth.

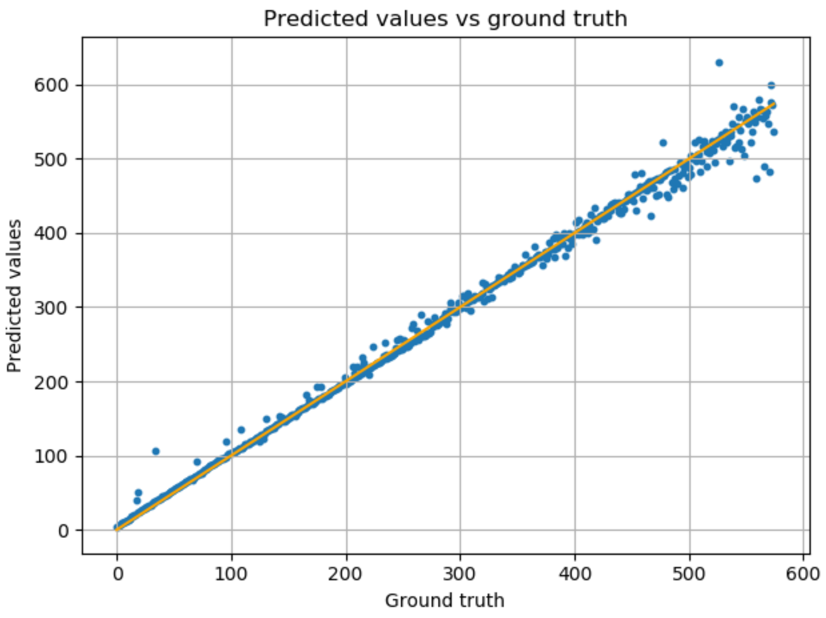


Figure 4.1: Predicted values vs ground truth in general models.

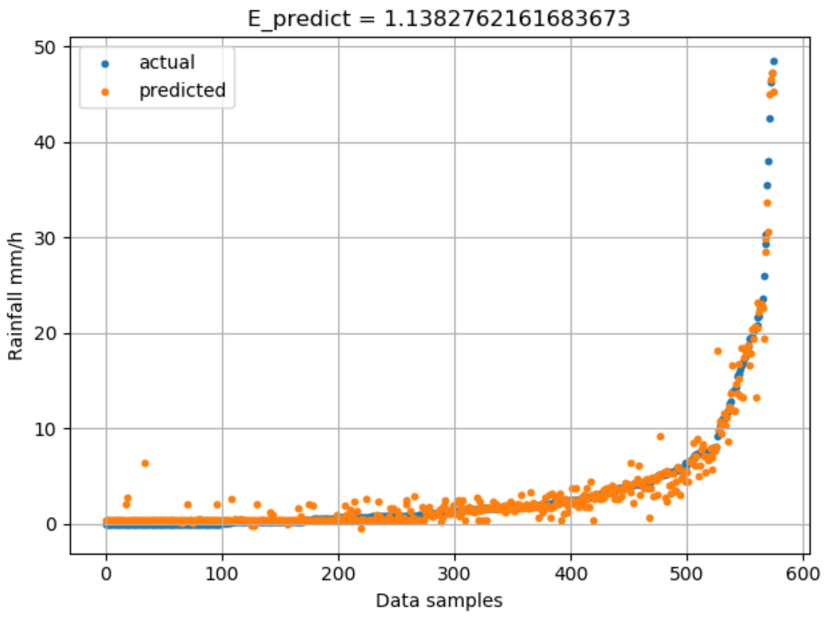


Figure 4.2: Predicted values vs ground truth in general models.

4.2 Comparison of architectures

As described in section 3.2, we compared 3 different models, varying mostly in network depth. Results are compared for specific models of selected stations. For this comparison we chose the best performing station and the worst performing station.

The models of stations with best performance achieved similar results, but the deeper model, the better results were recorded. The comparison of learning curves is displayed in the figure 4.3 and the comparison of performance graphs is shown in figures 4.4 and 4.5. You may notice a difference in number of epochs in each architecture, since *early stopping* method was used to avoid overfitting.

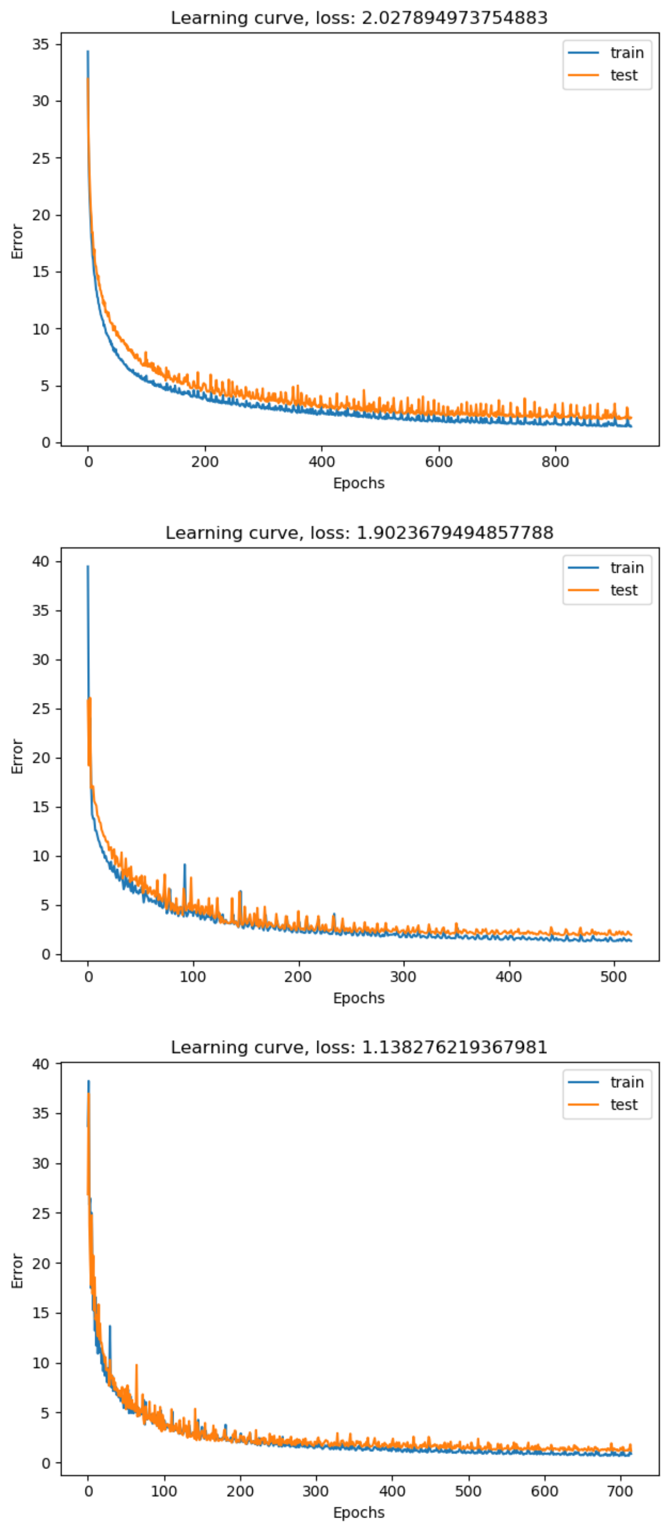


Figure 4.3: The comparison of shallow (top), medium (middle) and deep (bottom) model fitting learning curves on best performing station.

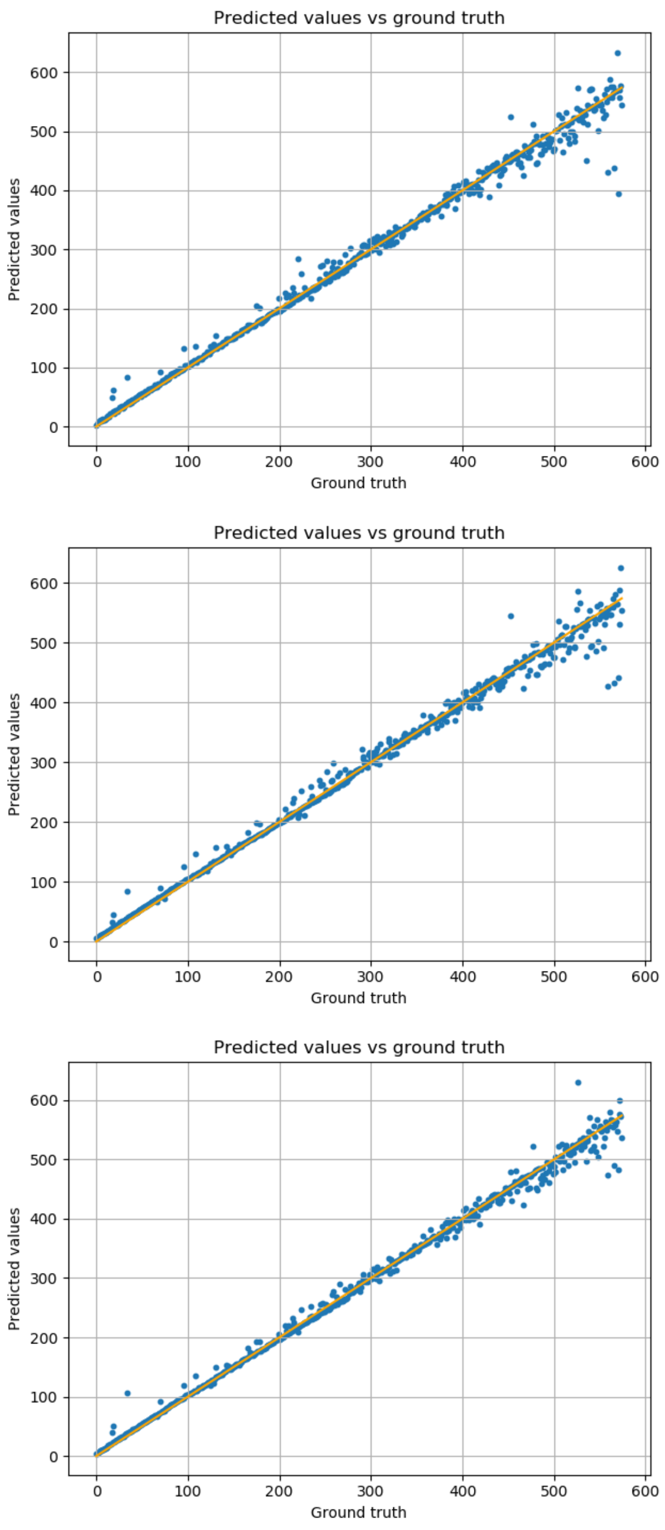


Figure 4.4: The comparison of shallow (top), medium (middle) and deep (bottom) model performance on best performing station.

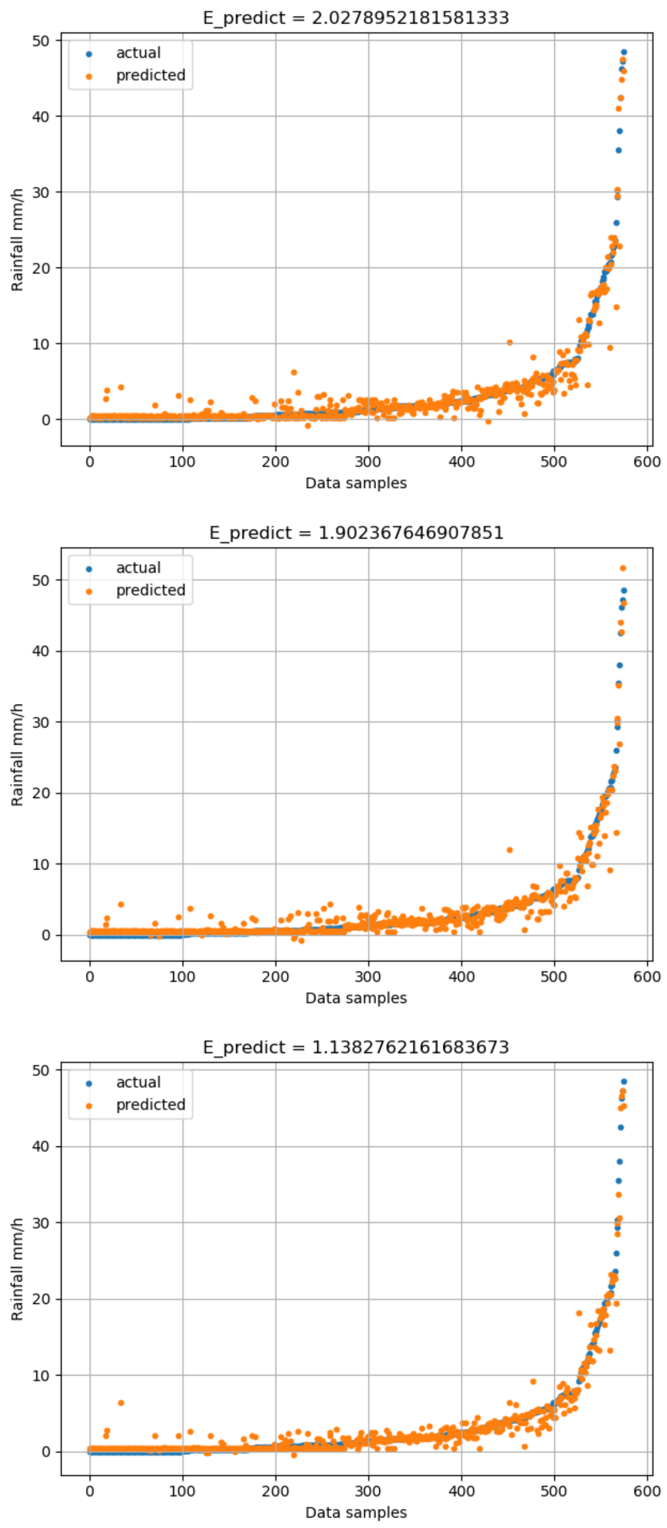
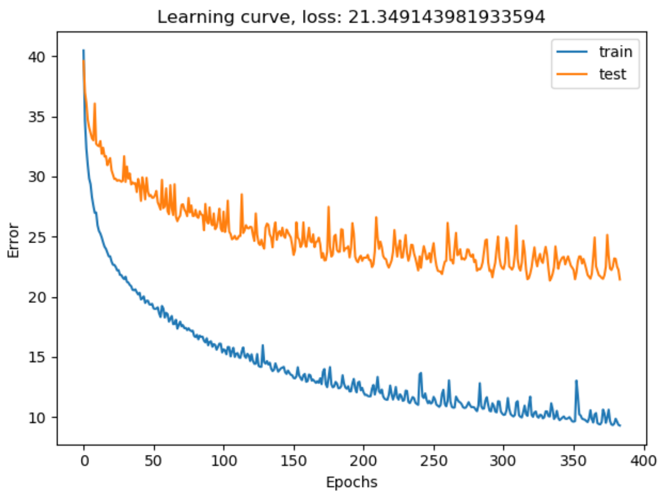
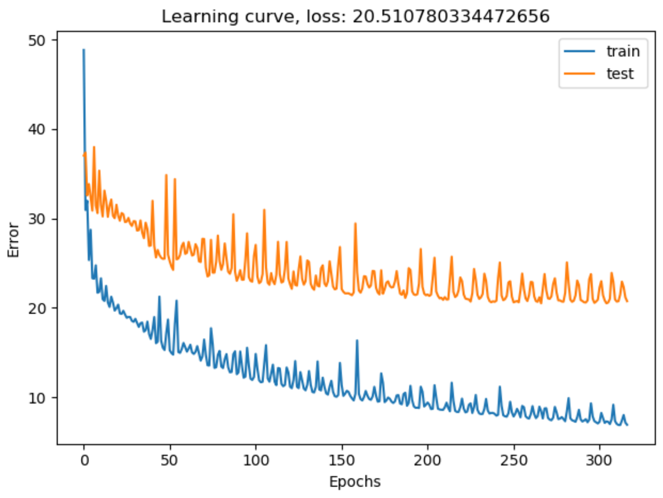


Figure 4.5: The comparison of shallow (top), medium (middle) and deep (bottom) model performance on best performing station.

The difference in performance of models of worst performing station was slightly more important.

However this step was only fine-tuning, since most of the improvement was made due to data preprocessing. Figures 4.6, 4.7 and 4.8 show the learning curves and performance graphs of worst performing station.





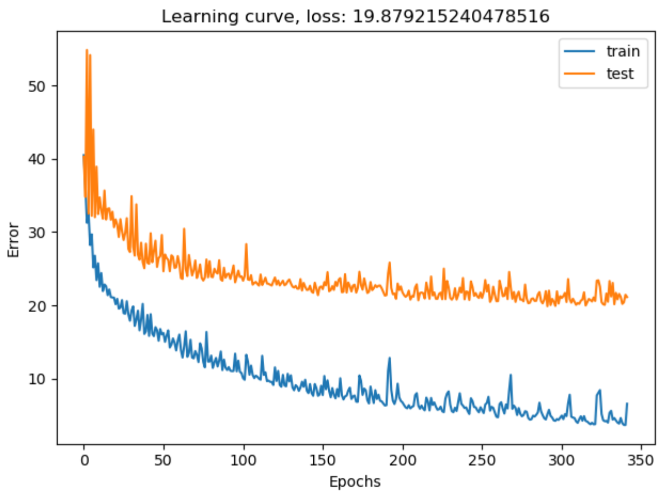


Figure 4.6: The comparison of shallow (top), medium (middle) and deep (bottom) model fitting learning curves on worst performing station.

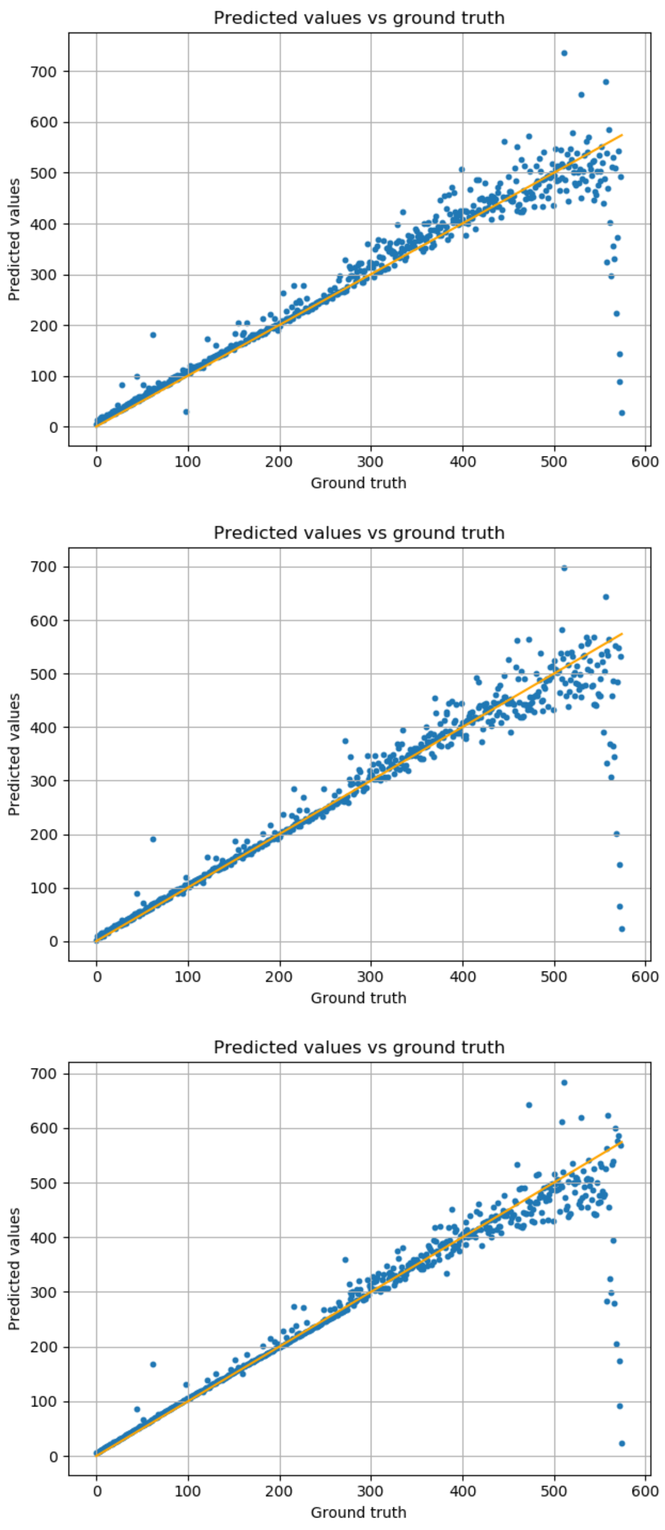


Figure 4.7: The comparison of shallow (top), medium (middle) and deep (bottom) model performance on worst performing station.

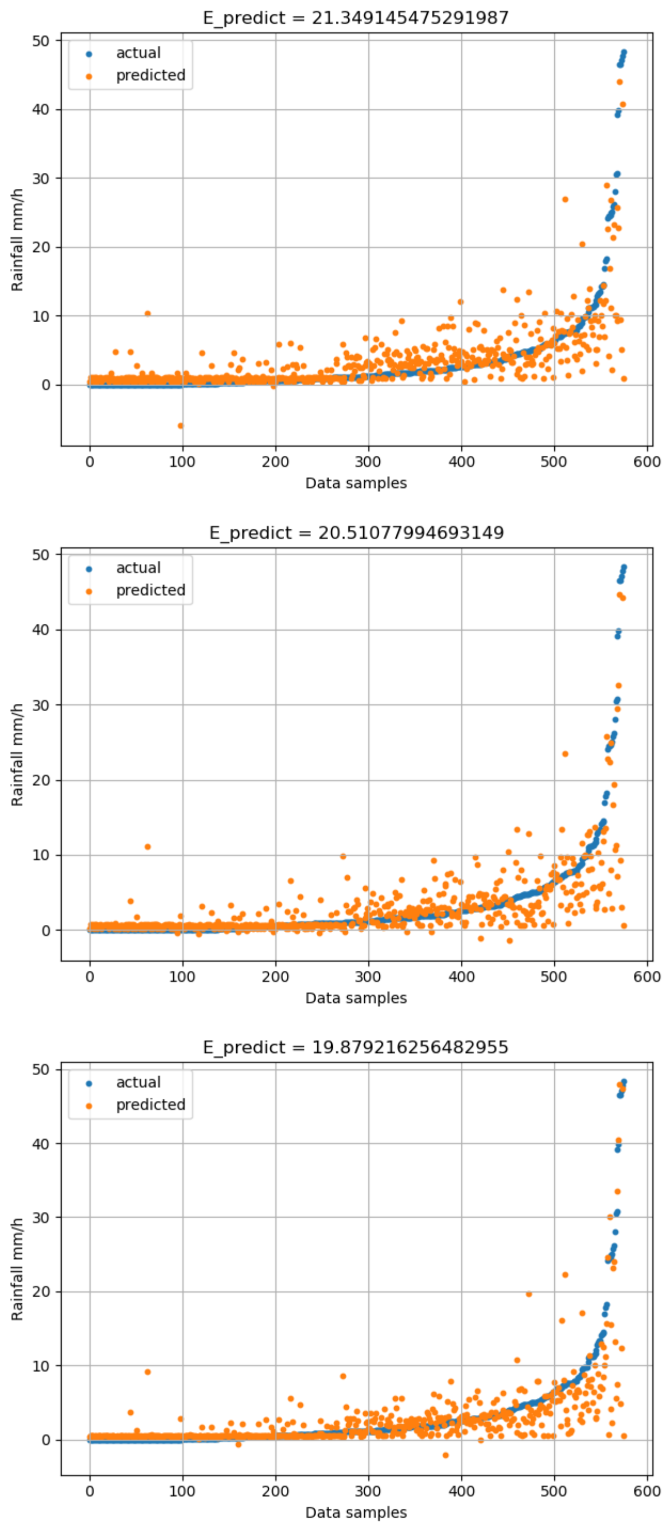


Figure 4.8: The comparison of shallow (top), medium (middle) and deep (bottom) model performance on worst performing station.

4.3 Specific vs general models

The experiment clearly demonstrated that the models for specific stations were superior to single general model.

The total of 137 models resulted in average of 7.9543 MSE, while the general model had a 25.4259 MSE. The figure 4.9 displays the histogram of specific models performance. The most commonly represented are values 3-10, meaning most of the models have an error lesser than $\sqrt{10}(\approx 3.33)$ mm per hour. The comparison of stations along with reference values are summed up in table 4.1 in the next section.

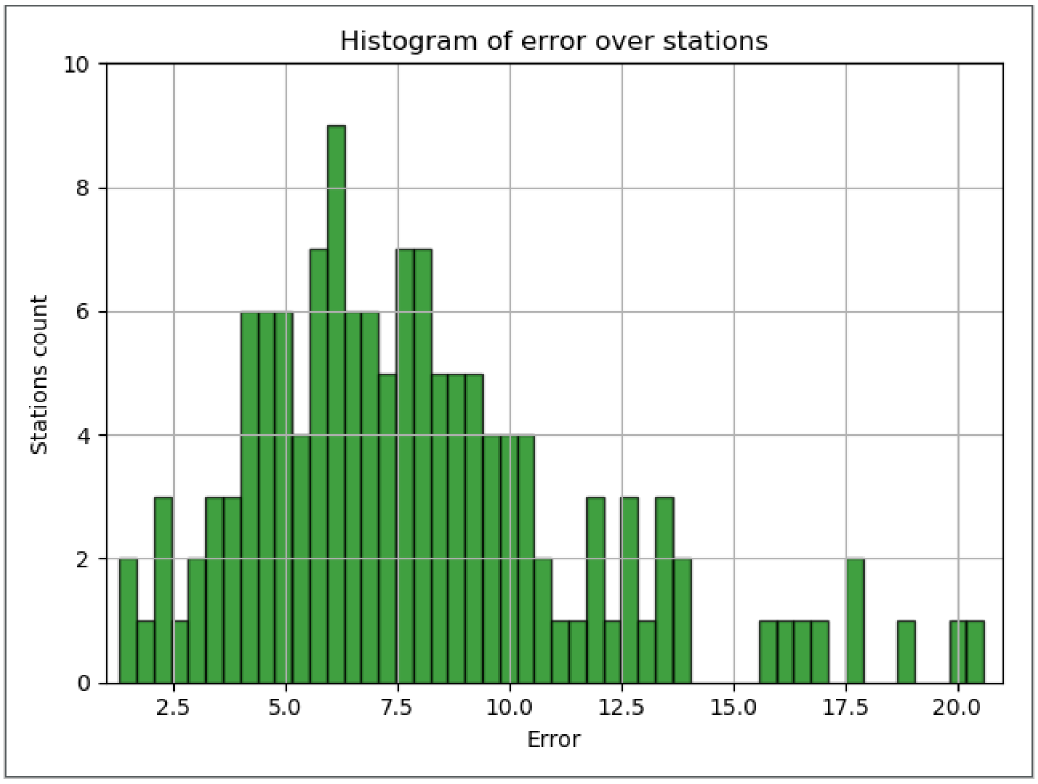


Figure 4.9: The histogram of MSE achieved by stations.

The figure 4.10 display the learning curve of general model fitting process and figures 4.11 and 4.12 show the performance evaluation of the model.

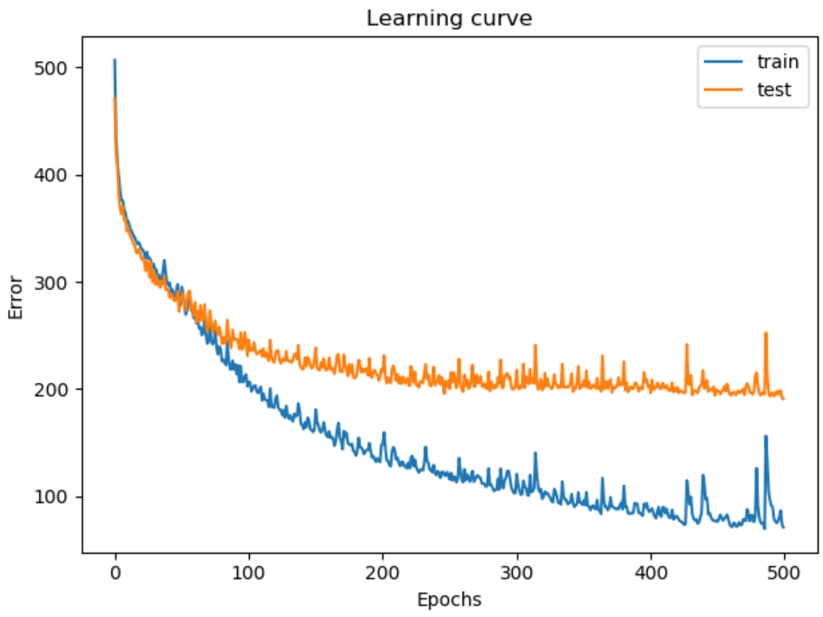


Figure 4.10: The learning curve of general model.

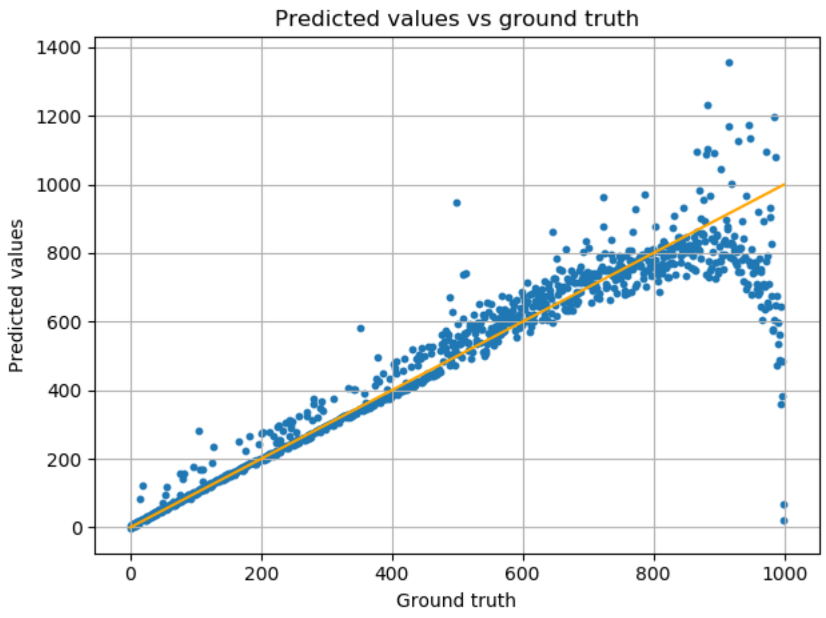


Figure 4.11: Predicted values vs ground truth in general models.

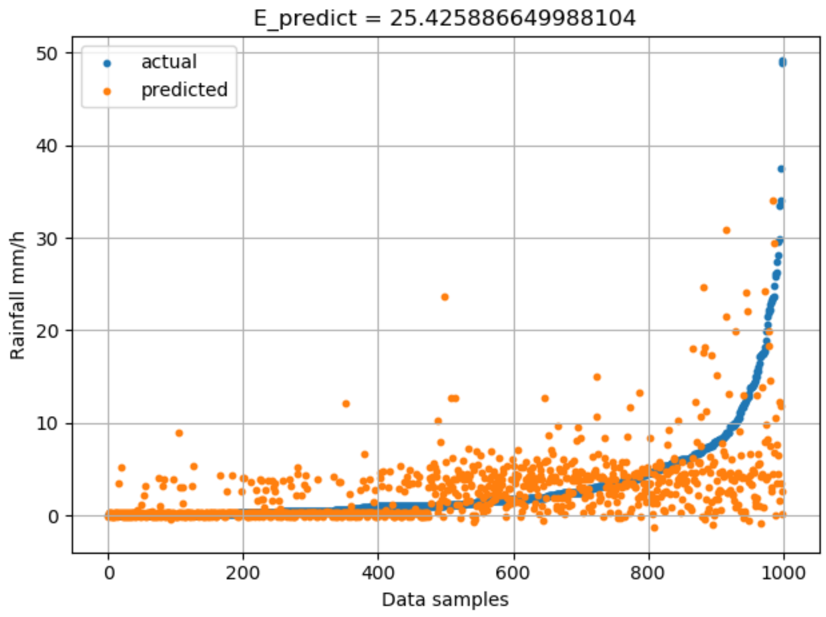


Figure 4.12: Predicted values vs ground truth in general models.

4.4 Reference values

As a final reference values, we used the values calculated using Marshall-Palmer distribution. However, for a single output value form a model, we have multiple reference values from measurements in different altitudes. A common way to approach this problem is selecting a maximal value out of the whole vertical profile. This interpretation is a slightly simplified version of results produced by SHMI.

The dataset with reference values added was generated later from the same data distribution as the train and test subsets, however results may slightly differ.

The figure 4.13 displays the histogram of reference values MSE distribution. You may notice, that some of the stations had immensely poor performance when it comes to reference values.

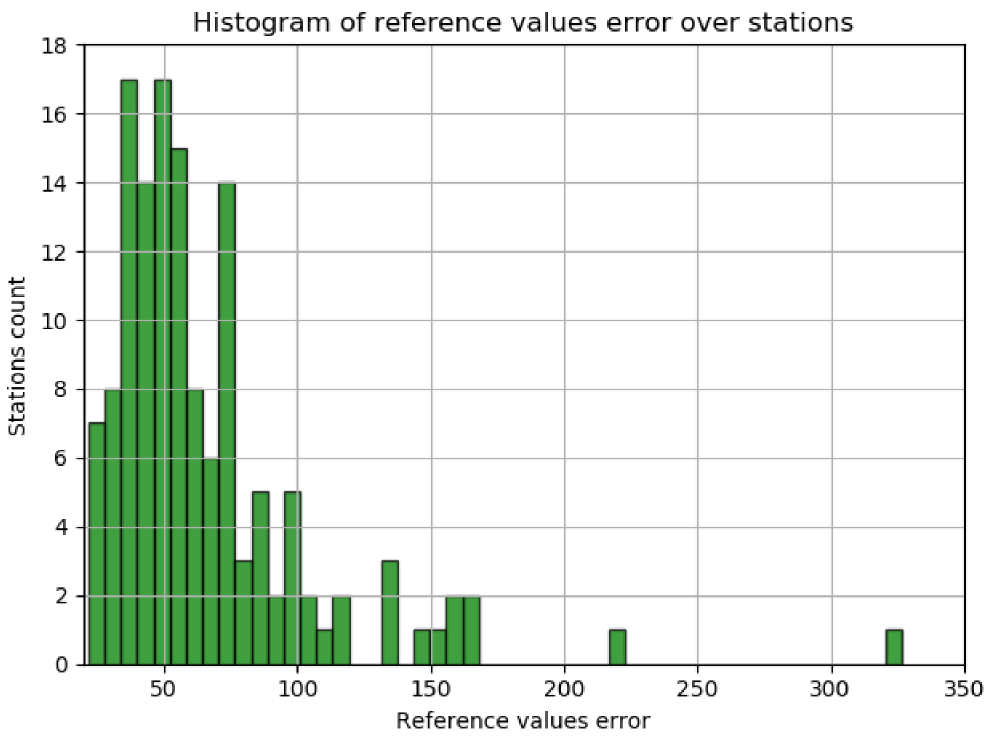
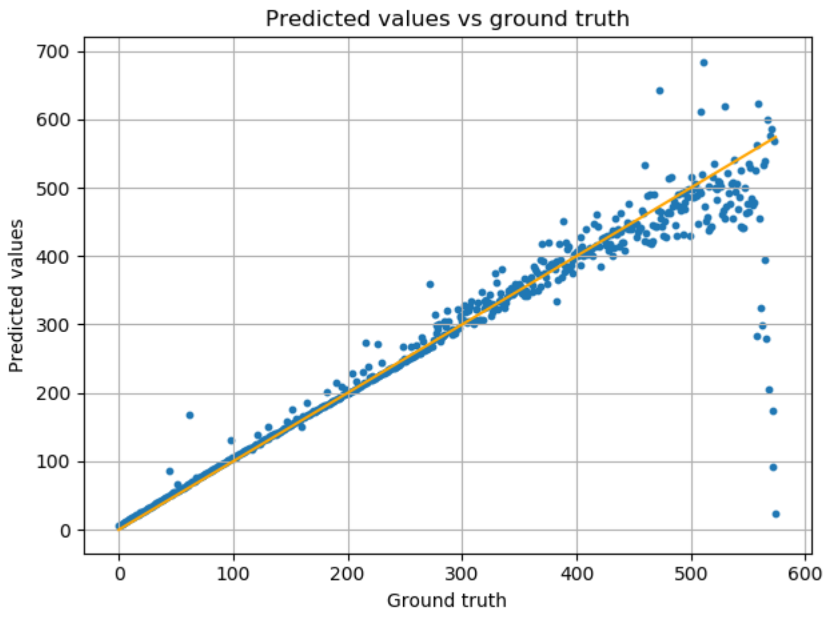


Figure 4.13: The histogram of MSE achieved by Marshall-Palmer values.

The comparison of the worst performing NN model is displayed in the figures 4.14 and 4.15. There is not so much of a difference in performance. The MSE of neural network is approximately half (model 19.88 vs reference 39.78 MSE) of the reference values, which is caused mostly by a small number of significant errors.

However, the great difference of performance is comprehensible in the comparison of the best performing model. Figures 4.16 and 4.17 display the comparison of the performance graphs of the station with best performance from models. The neural network model performs approximately 146 times better when it comes to MSE metric. You can clearly see that model predictions are superior to reference values in the graphs.



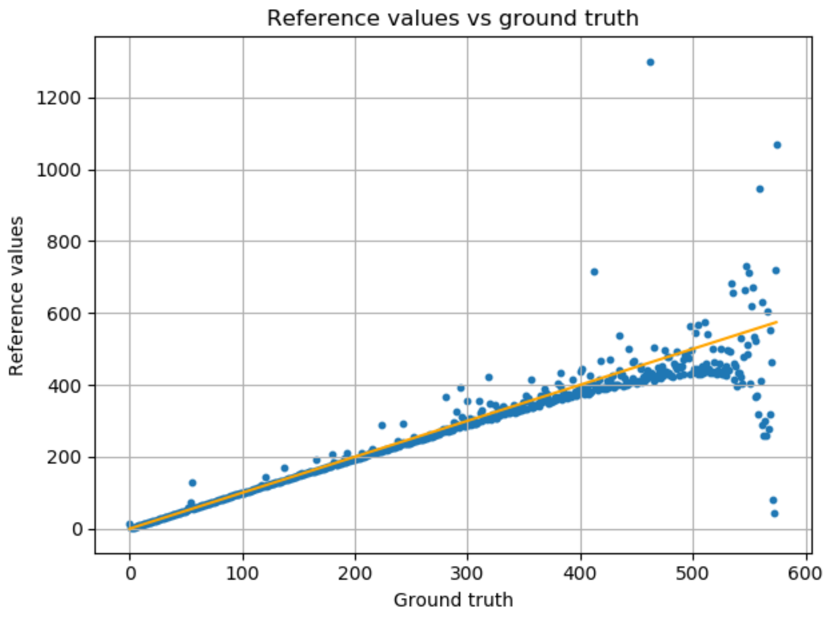
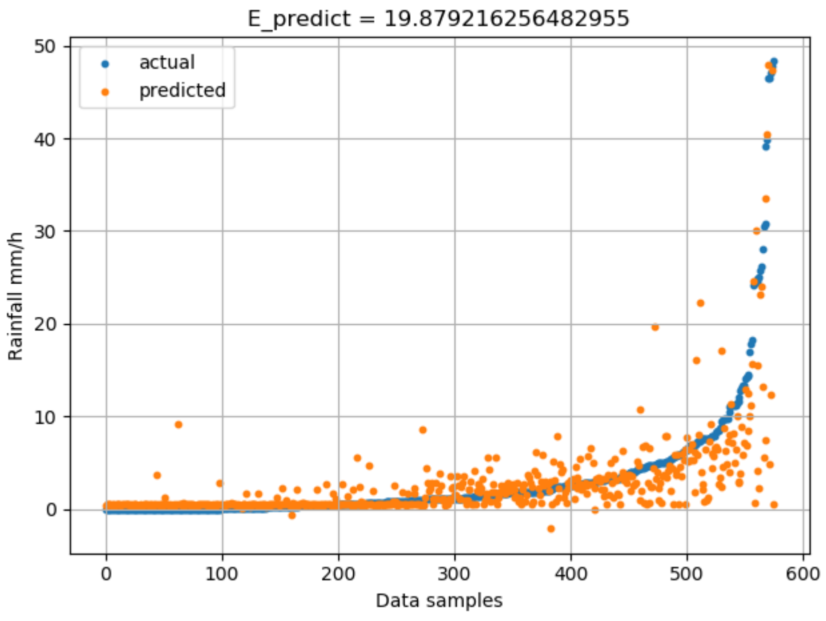


Figure 4.14: The comparison of predicted values (top) and reference values (bottom) versus ground truth on best performing station.



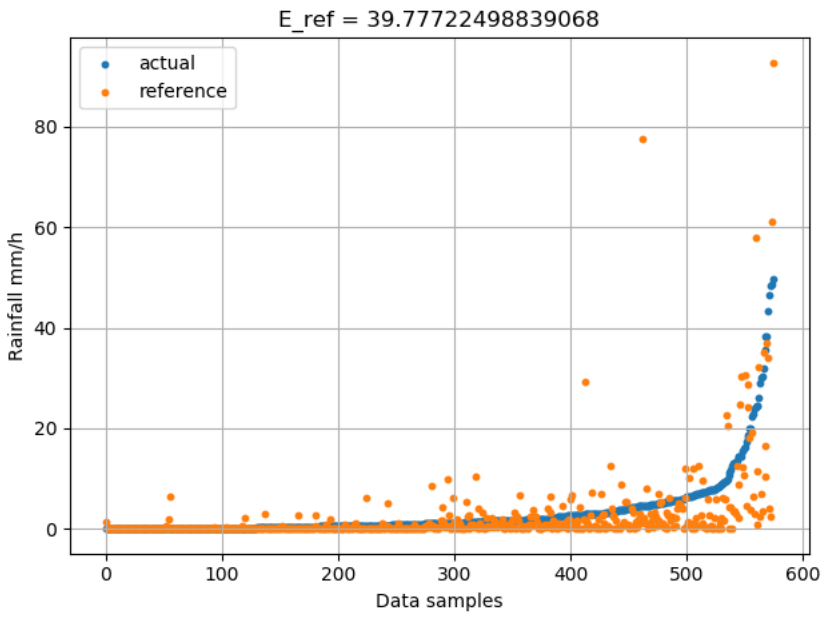
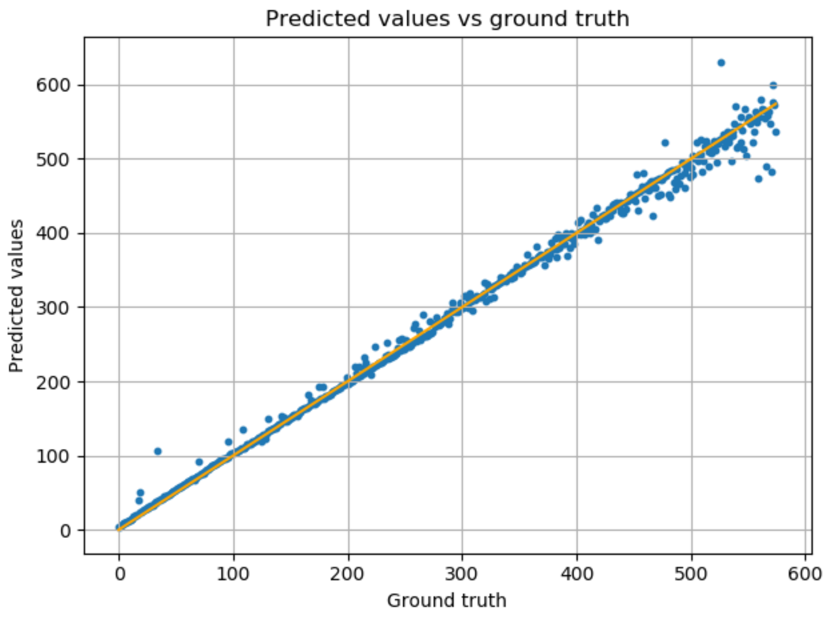


Figure 4.15: The comparison of predicted values (top) and reference values (bottom) versus ground truth on best performing station.



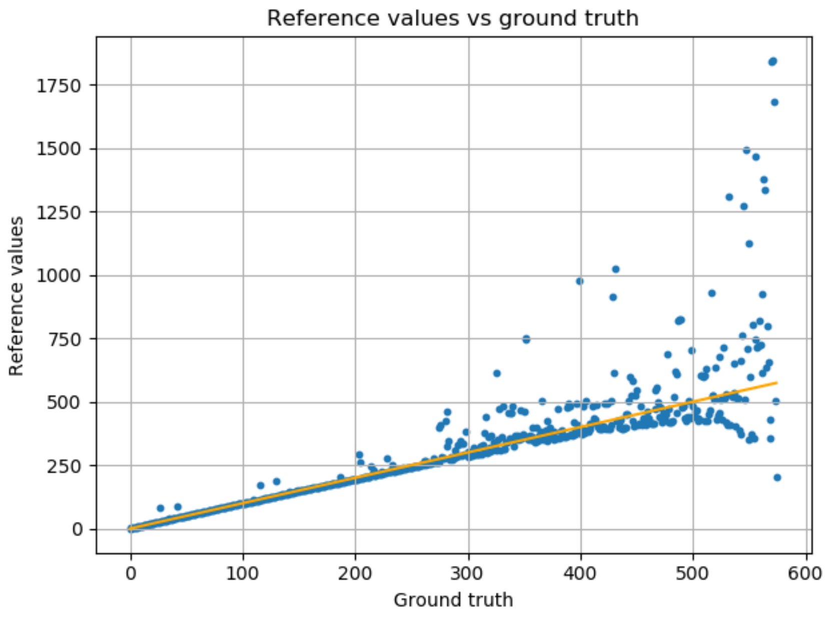
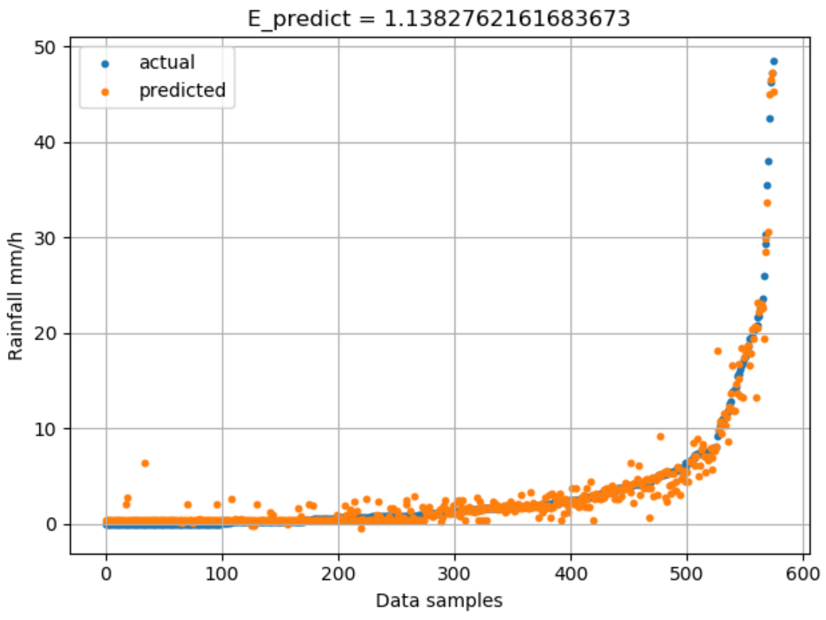


Figure 4.16: The comparison of predicted values (top) and reference values (bottom) versus ground truth on best performing station.



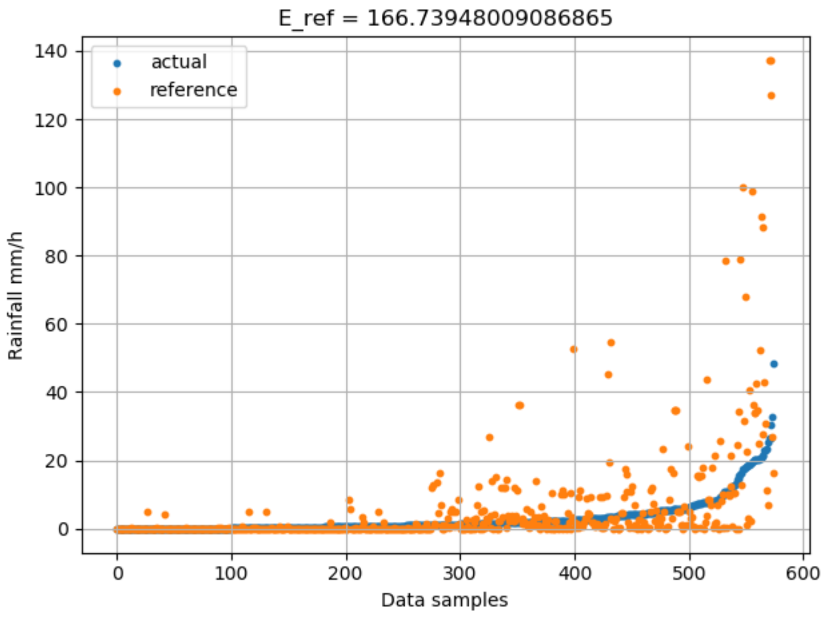


Figure 4.17: The comparison of predicted values (top) and reference values (bottom) versus ground truth on best performing station.

The table 4.1 sums up the performance comparison of specific models, general model and Marshall-Palmer values thought of as reference values.

The table shows that the performance of models for specific stations are superior to other results.

Table 4.1: The comparison of some specific, general models and reference values.

Model	MSE	Station id
best specific	1.1383	42137
average specific	7.9543	x
worst specific	19.8792	42051
general	25.4259	x
best reference	21.7827	42013
average refrence	65.8855	x
worst reference	326.625	42043

Chapter 5

Conclusion

The goal of this thesis was to create an artificial neural network model, which would be superior to results achieved by SHMI using a Marshall-Palmer relation.

We created multiple recurrent models capable of achieving such results. Model architectures were inspired by Aaron Sim's work in 2015, which won a data science competition. Results presented in chapter 4, Results and Evaluation, clearly show that model created for the purpose of the thesis is superior to calculations obtained by using Marshall-Palmer. In average, our models achieved over 8 times better results than results achieved by SHMI using Marshall-Palmer relation. The main goal of the thesis was successfully accomplished.

Major aspects of creating a well-performing model was utilizing the recurrence in neural network and a great deal of data preprocessing for the needs of recurrent network. The thesis justified the applicability of recurrent neural networks for certain type of task, such as dealing with numeric data in time series.

The future work of the thesis lies in further researching parameters and architectures of networks. The goal would be either to replace multiple models for all of the stations by single fine tuned and well fitted model or to approach each specific model individually and improve its performance independently to other models. We also discussed possibility of creating specific models for different seasons or even months, since various weather features might differ.

Bibliography

[BGV92] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classier. *Proceedings of the fifth annual workshop on Computational learning theory – COLT '92. p. 144*, 1992.

[Bli11] Vojtěch Bližňák. *The exploitation of remote sensing measurements for the analysis and development of rainfalls.* PhD dissertation, Charles University in Prague, Faculty of Science, 2011.

[Bro17] Jason Brownlee. Gentle introduction to models for sequence prediction with recurrent neural networks. July 2017.

[Elm90] J. L. Elman. Finding structure in time. *Cognitive Science, 14: 179-211*, 1990.

[Hay09] Simon Haykin. *Neural Networks and Learning Machines.* Pearson Education Ltd., 2009.

[JKOM14] M. Jurasek, J. Kanak, L. Okon, and L. Meri. Close future of slovak weather radar network. *ERAD 2014 - THE EIGHTH EUROPEAN CONFERENCE ON RADAR IN METEOROLOGY AND HYDROLOGY*, 2014.

[Jor89] M. I. Jordan. Serial order: A parallel distributed processing approach. 1989.

[KBP$^+$97] V. Kvasnička, Ľ. Beňušková, J. Pospíchal, I. Farkaš, P. Tiňo, and A. Kráľ. *Úvod do teórie neurónových sietí.* Iris, 1997.

[Kuz14] Tomáš Kuzma. *Deep Learning in Neural Networks.* PhD dissertation, Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics, 2014.

[LJH15] Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. A simple way to initialize recurrent networks of rectified linear units. April 2015.

[MP43] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mothemnticnl Biology Vol. 52, No. l/2. pp. 99-115. 1990*, 1943.

[MP69] M. Minsky and S. Papert. *Perceptrons: an introduction to computational geometry.* MIT Press, Cambridge, MA, 1969.

[Sim15] Aaron Sim. Estimating rainfall from weather radar readings using recurrent neural networks. December 2015.