Short-Term Quantitative Precipitation Forecasting using Radar Data and Neural Networks

Luisa Fernanda Castaño Jurado (4808906)
luisa.castano@fu-berlin.de

A thesis presented for the degree of
Bachelor of Science

Supervisor:	Paras Mehta
First Advisor:	Prof. Dr. Agnès Voisard
Second Advisor:	Prof. Dr. Tim Landgraf

Freie Universität Berlin
Department of Computer Science & Mathematics
Institute for Computer Science
Research Group: Databases and Information Systems

October 19th, 2017
Berlin, Germany

Abstract

The short-term prediction of rainfall based on radar data is an important part of meteorology with uses that vary from flood prevention through traffic control to event planning. Modern systems allow for highly accurate approximation of the probability of rain within 6 hours, but not yet for the specific amount. The goal of the CIKM AnalytiCup 2017 is to improve existing systems to predict liquid precipitation quantities with 1 hour lead time by using exactly 1.5 hours of accumulated radar data.

Existing systems use computer vision algorithms to track convective cells and an empirical mathematical relation to convert the reflectivity values of these to rain rates, aside from complex physical simulations. Furthermore, they have more information available, such as wind direction and speed, temperature, atmospheric pressure, and topography. The first goal of this thesis is to solely use radar echo extrapolation data to achieve predictions with a RMSE lower than 14.69 mm/h, the second goal is to explore the performance of the statistical model used against a state-of-the-art Quantitative Precipitation Forecasting (QPF) system. Since the relationship between reflectivity values and the amount of rain is strongly non-linear, and given that artificial neural networks (ANN) are able to approximate any function in $\mathbb{R}^n$, this thesis evaluated different reknown architectures of Convolutional Neural Networks as predictions models for the described problem. Although the set baseline was not reached, the comparison against a chosen industry-based QPF left a promising indicator for further research.

Zusammenfassung

Die auf Radardaten basierte kurzfristige Vorhersage von Regenmengen ist ein wichtiger Bereich der Metheorologie, dessen Auswirkungen von Hochwasserschutz über Verkehrskontrolle bis zu Veranstaltungsplanung betreffen. Moderne Systeme erstellen hoch akkurate Einschätzungen der Regenwahrscheinlichkeit innerhalb 6 Stunden, aber noch nicht der genauen Quantität. Das Ziel der CIKM AnalytiCup 2017 ist, die existierende Systeme für Vorhersage mit 1 Std. Vorlaufzeit von flüßigen Niederschalgsmengen zu verbessern unter der Nutzung von 1.5 Stunden akkummuliereten Radardaten.

Existierende Systeme verwenden neben komplexen physikalischen Simulationen, Computer Vision Algorithmen um Konvektionszellen zu verfolgen, und eine mathematische Relation (Marshall & Palmer) um die Reflektionswerte dieser in Regenraten umzuwandeln. Darüber hinaus, verfügen diese Systeme über mehr Information, wie zum Beispiel Windgeschwindigkeit und -richtung, Temperatur, Luftdruck, und Topographie. Das erste Ziel dieser Arbeit ist die Vorhersage von Regenmengen mit einem RMSE besser als 14.69 mm/h, nur unter Verwendung von Radar Echo Extrapolationsdaten. Das zweite Ziel der Arbeit ist die Evaluation des erstellten Vorhersagemodells im Vergleich mit einem state-of-the-art Quantitative Precipitation Forecasting (QPF) System. Da die Relation zwischen Reflektionswerten und Regenmenge stark nicht-linear ist und Artifizielle Neurale Netze (ANN) jede Funktion in $\mathbb{R}^n$ approximieren können, wurden in dieser Arbeit verschiedene etablierte Convolutional Neural Network (CNN) Architekturen als Vorhersagemodelle für das beschriebene Problem evaluiert. Obwohl die vorgegebene Performance-Baseline nicht erreicht wurde, stellen die Ergebnisse des Vergleichs mit dem professionellen QPF-System vielversprechendes weiteres Forschungspotenzial in Aussicht.

Declaration of Honor

I do solemnly declare that I prepared this thesis independently and that the thoughts taken directly or indirectly from other sources are indicated accordingly. The work has not been submitted to any other examination authority and also not yet been published.

October 19th, 2017

Luisa Fernanda Castaño Jurado

Table of Contents

List of Figures

Chapter 1

Introduction

The current chapter gives an overview of the motivation to improve the short-term prediction of rain quantities, the goals of this thesis, and the structure of the work.

1.1 Motivation

Modern weather systems can predict precipitation probabilities with high accuracy, but not knowing the approximate amount of rainfall means not being able to activate prevention protocols when this amount reaches certain thresholds. The first type of flood that severe rain can cause is the pluvial one, or surface flood. This happens when heavy precipitation saturates the urban drainage system, having water flow into the streets as a result. Water from nearby hills, run-off, can also flow into cities, especially after recent forest fires. The second type of flood associated to rain is the fluvial one, which occurs when water overflows over the edge of rivers. Causes leading to this are intense rain accompanied by runoff (the water, derived from precipitation, that ultimately reaches stream channels [33]) flowing into the river or stream. Forest fires also contribute to this natural disaster by facilitating that more sediments engross the volume of the river.

Runoff simulations are an essential method to predict these types of events. They are carried out with measurements of rainfall in rain gauges. The problem is that rain recorders are sparse on open field and interpolation of precipitation value lead to errors.

Summarized, live accurate information on short-term precipitation amounts can be crucial to act quickly and prevent disasters on both on the short (thunderstorms) and the long term (fluvial floods).

As an example, Berlin had severe precipitation on June 29th and 30th, 2017. The maximum report in the metropolitan area was 197 l/m^2. In Figure 1.1 we can see that the forecast 24 hours before severe rain on June 29th is highly inaccurate, at least for Berlin. The Severe Event Catalogue of the European Center for Medium-Range Weather Forecasting (ECMWF) [1] contains diverse examples of failures to foresee heavy precipitations, a sign that quantification of rain is a field where research can still lead to considerable improvement. It is worth mentioning that these incidents are based on their medium-range forecasting systems, which make predictions with a $\geq$ 24h lead time. According to the World Meteorological Organization (WMO), more appropriate systems for nowcasting (prediction with $\leq$ 24h lead time) are being developed since 2011, primarily in Germany and France [27]. These forecasting models are not fully implemented in production, which is confirmed by the fact that the Verification Report of the Performance of Local Weather Forecasting and Warnings of the Winter 2016/2017 carried out by the German Weather Service does not perform an evaluation of rainfall predictions, it only does so on the probability of precipitation. [31]

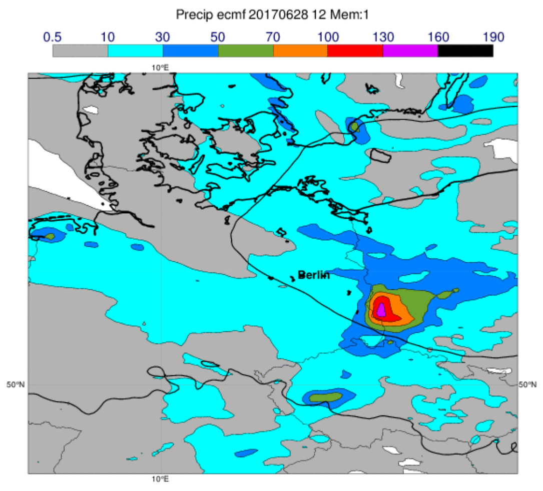


Figure 1.1: The plot shows 24-hour precipitation valid 29 June 06H to 30 June 06H from HRES (High-RESolution) forecasts measured on June 28th at 12h. The scale is given in mm. Excerpt from [30].

[1]https://software.ecmwf.int/wiki/display/FCST/Severe+Event+Catalogue, accessed August 29th, 2017

1.2 Goal

The goal of this thesis is to develop a short-term Quantitative Precipitation Forecasting (QPF) system using Artificial Neural Networks (ANNs) as a training model. More precisely, this objective can be subdivided into two subgoals:

First, this thesis has been written in context of the Conference on Information and Knowledge Management (CIKM) AnalytiCup 2017 contest. Hence, the immediate aim of it is to develop a model that can predict the amount of rainfall in the future 1 to 2 hours with a root mean-squared error (RMSE) lower than 14.69 mm/h. This metric is the one achieved by the organizers of the contest with linear regression [1]. The RMSE is a measurement of accuracy for prediction errors that is sensitive to outliers, so it can be inferred that the underlying goal of the contest is to be able to identify data that leads to extraordinary heavy precipitation, the one for which both public institutions and individuals want to be prepared for.

.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2}$$

Root-Mean Squared Error, performance metric for regression models where y_i is the real value for dataset i, y_i' is the predicted value, and n is the size of the sample.

Based on the system developed for the contest, the second goal of this thesis is to evaluate whether ANNs can achieve similar results as those reached by highly sophisticated QPF systems used in the meteorological industry. If this is the case, should further research consider the integration of ANNs in QPF systems?

As we will see in the second chapter, although ANNs have gained success in the last couple of years, their application to weather forecasting has been sparse. This is probably due to the fact that the focus of the research about low-range prediction systems has been on improving the complex physical models by applying learnings of chaos theory, increased computational power, pattern matching and computer vision. Summarized, this thesis does not expect to beat the metrics of meteorological services, but hopefully find reasons to assert that Machine Learning can be applied to forecasting with success.

1.3 Structure of the thesis

The remainder of this thesis is structured as follows:

Chapter 2 deals with the background and the context of the thesis. Moreover, it presents currently used methods for Quantitative Precipitation Forecasting (QPF), among which is Scout, a forecasting system used in the industry that will be used to compare the performance of the model developed in the margins of this thesis.

Chapter 3 describes what the data looks like and explains the logical steps that led to the chosen tools. It also contains a general introduction to neural networks and typical challenges faced while working with them.

Chapter 4 displays the evolution of finding the right architecture for the neural network, as well as the application of an optical flow algorithm to find motion in the data. The comparison of the performance of the developed model and Scout will take place in this chapter.

Chapter 5 summarizes the work done and reflects on the limitations and lessons learned, while setting the guidelines for future work.

The last chapter contains a list of all referenced books, scientific papers and articles which are relevant for the thesis.

Chapter 2

Related Work

This chapter lists all special definitions needed to understand the background of this work as well as scientific work with findings relevant to this thesis.

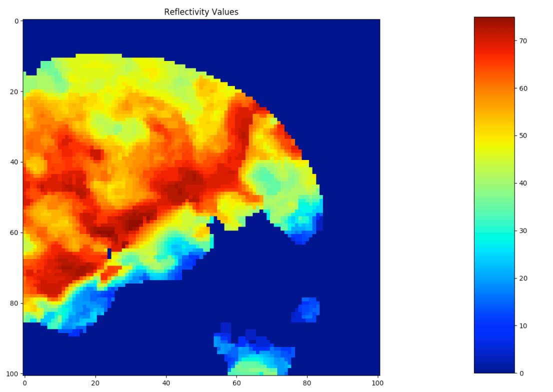
2.1 Background on Radar Echo Extrapolation Data

This kind of data is collected when a weather radar emits electromagnetic waves in short pulses in a defined direction. The proportion reflected off hydrometeors is received by the same radar unit. Missing data is extrapolated using surrounding echoes [26].

> Hydrometeor: "Liquid particles formed in the atmosphere such as fog, clouds, raindrops, and solid particles such as ice, crystals, and snow." [26]

The intensity with which the wave is reflected is called *reflectivity value Z* and it is measured in decibels to Z, dBZ [26]. The radar used for this thesis emitted waves every six minutes. The radius of the waves is not known, given that the location of the radar was not published to avoid the contestants using topographical and historical data to improve their predictions.

The CIKM AnalytiCup 2017 has given us the Cartesian data with 1 km^2 resolution, meaning that each value corresponds to the Z value of 1 km^2 in a total area of 101 $\times$ 101 km^2, resulting in a matrix of 101 $\times$ 101 cells detailing the state of the hydrometeors in that area of the atmosphere. It is possible to visualize each km^2 as a pixel and the mentioned matrix as a picture looking like Figure 2.1. We also have four such images which were *taken* every six minutes over a period of 1.5 hours, so 60 overall. Each one of the four images corresponds to a different height of the atmosphere: 0.5, 1.5, 2.5, and 3.5 km. Plotted all together in Figure 2.1 according to time and height, it is possible to recognize motion and direction. Moreover, it is possible to visualize convective cells, which are clusters with reflectivity values higher than 35 dBZ covering more than 4 km^2. The lifespan of these cells is less than one hour and they are responsible for the formation of heavy precipitation [29]. Radar data with a resolution lower than 4 km^2 is not appropriate to observe the evolution of convective systems.



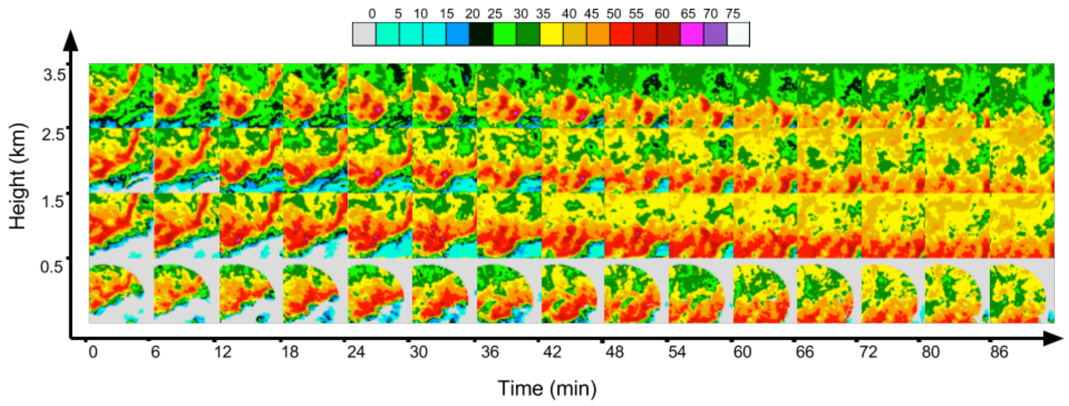


Figure 2.1: Above: Area of 101 × 101 km^2 reflectivity values. Below: Each 101 × 101 km^2 image according to time and height.

For rainfall forecasting purposes, each Z value can be translated into an intensity rate using the Marshall-Palmer relationship between the reflectivity value Z (dBZ) and the rain intensity R (mm/h)

$$R = aZ^b$$

where a and b are values interpolated from a large amount of measured drop-size distributions [23]. Since this interpolation is only possible post-precipitation, recommended values by Marshall and Palmer are $a = 200, b = 1.6$ [23]. However, the Association of Engineers shows in [26] that different interpolated values for a and b result in differences of up to 119 mm/h for the same Z value. This is one of the points of improvement within Quantitative Precipitation Forecasting (QPF).

Radar Echo Extrapolation Data is the kern of the state-of-the-art methods currently used for precipitation forecasting, especially for QPFs. These methods are called Ensemble Prediction Systems (EPS) [27] and one of the biggest computational centers in charge of creating the Numerical Weather Prediction (NWP) systems that form the ensembles is the European Center for Medium-Range Weather Forecasts (ECMWF) [13]. The global, regional, and local models generated by this center are shared with the meteorological centers of the 34 cooperating members, among which is the German Weather Service. The next sections contain a brief introduction to Ensemble Prediction Systems with a concrete example.

2.2 WMO Ensemble Prediction Systems

The general trend in meteorology and weather forecast models is the use of Ensemble Prediction Systems (EPS), which are "numerical weather prediction systems that allow us to estimate the uncertainty of a weather forecast as well as most of the most likely outcome. Instead of running the model once, the model is run many times from very slightly different initial conditions." [27, p.2]. The reason for the need of ensembles is that the processes of the atmosphere can be explained by the chaos theory, making it highly sensitive to small changes and meaning that differences in initial parameters of single models can cause wrong forecasts. Ensemble forecasts perturb the models very slightly and rerun them many times. The certainty of the final forecast is directly proportional to the number of end-simulations with the same result. The WMO differentiates three EPSs, the global one (3-15 days forecasting), the regional EPS (1-3 days head time), and the convective-scale EPSs, which are the ones applied to nowcasting, the predictions to events happening within the next 24 hours. This is the type of forecasting the applies to our problem: rainfall prediction within one hour. Convective NWPs have a resolution between 1 and 4km that allows concentrating on the intensity of, for example, thunderstorms. As a downside, models change fast and forecasts can rapidly become invalid. As of 2011, the experience with convective EPS was very limited. One example is the COSMO-DE system, used by the German Weather Service.

2.2.1 Quantitative Precipitation Forecasting with COSMO-DE

COSMO-DE is a regional model that covers Germany, as well of parts of Austria, Switzerland, and of other neighboring countries. Its horizontal resolution is 2.8 km and it has 50 vertical layers. Even though it is not a convective model, its low resolution allows the observation of convective events, meaning that this model can be also used to generate Quantitative Precipitation Forecasts. COSMO makes a prediction for the next 27 hours every three hours. The input data is delivered by the network of weather radar stations across Germany. The first step is the assimilation of data which is a list of quality control measures applied to the radar data, including elimination of ground clutter and anomalous propagation. Given that SCOUT, our comparison model, implements the same quality control, each step of it will be further discussed in the next section [36]. Important within the data assimilation of the COSMO-DE is the *latent heat nudging* approach: a model is created based on the observations of each radar in the highly overlapped mesh seen in Figure 2.2, then, each grid point of the model is compared to the corresponding radar grid point and the difference is used to enhance the vertical profile of modelled latent heat at that point. This is a relevant improvement because the assumed relation between precipitation formation and latent heat release is used to adjust the model dynamics and produce better predictions [19]. Once the data is assimilated, a simulation attaining to highly complex physical events that escape the scope of this thesis is run to forecast the state of the atmosphere within the next 27 hours. The rain intensity for each grid point is finally calculated using the Marshall-Palmer relation with values a and b calculated based on the development of the Z values for each given point of the mesh. The German Weather Service has not calculated numerical verification metrics for the rainfall predictions of the COSMO-DE model, so a comparison with it is not possible. Instead, this thesis will consider the SCOUT system for said performance comparison.

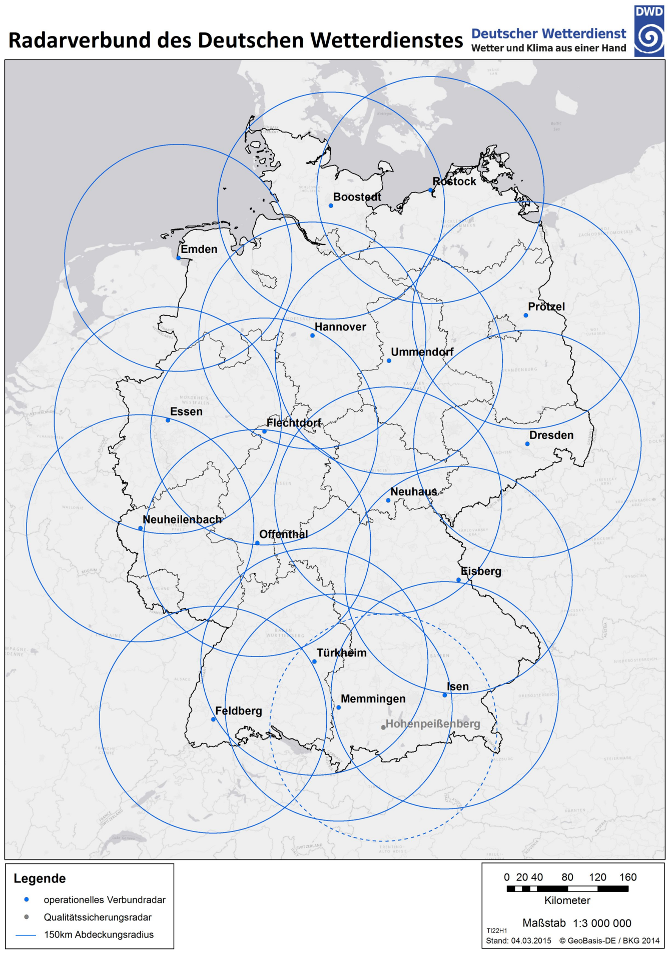


Figure 2.2: Radar Network of the German Weather Service. Copyright 2015 GeoBasis-DE/BKG 2014. Available online at: www.dwd.de/DE/derdwd/messnetz/atmosphaerenbeobachtung/_functions/Teasergroup/radarverbund_Karte.jpg, accessed September 2th, 2017

2.3 Radar Processing System SCOUT

The radar processing system SCOUT uses radar and rain gauges data as well as information on numerical weather models. This system offers detailed rain measurements as well as up to three-hour rain predictions based on radar data. SCOUT is used in Germany, Switzerland, the Netherlands, and Thailand. In order to carry out a radar measurement or a make a prediction, SCOUT first does a quality control on the input data then a pattern-matching analysis to model the future radar images for that area and then a Marshal-Palmer conversion to predict the rain intensities.

2.3.1 Data Quality Control

The data collected by a radar is affected by perturbations such as beam blockage, clutter, ground echoes, beam attenuation, or noise, and they need to pay attention to the difference between convective (fast and intense) and stratiform (mild and continuous) precipitation. Below are shown some short examples of how SCOUT deals with the mentioned phenomena.

Beam blockage Beam blockage can happen due to the presence of buildings or mountains. This can be corrected using a digital elevation model, but these are not always available for every radar, which is why SCOUT does not use one for the correction. Instead, it adds up the reflectivity values of each beam and uses the comparison among neighboring degrees to correct blockages. Figure 2.3 shows the before and after images of such correction.

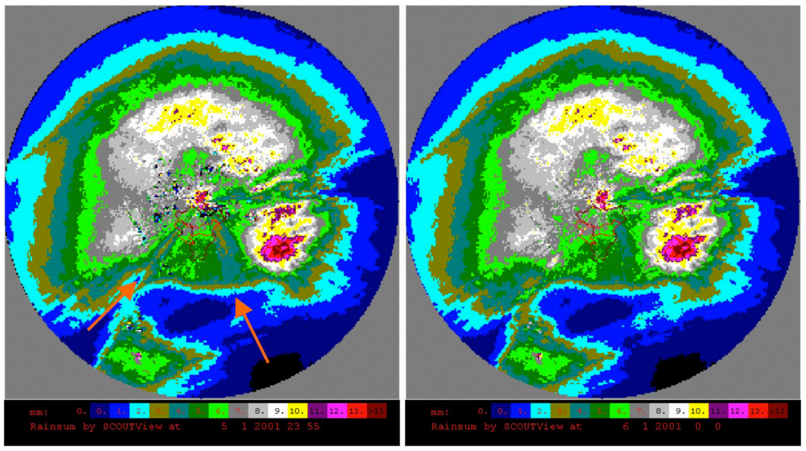

mm: 0. 0. 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. >13
Rainsum by SCOUTView at 5 1 2001 23 55
mm: 0. 0. 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. >13
Rainsum by SCOUTView at 6 1 2001 0 0

Figure 2.3: Radar picture (Essen) before and after the correction in the south-east and west. The orange arrows point to blocked beams. Image extracted from [5].

Clutter and Ground Echo The clutter removal step tries to eliminate pixels that are not associated with rain. Causes for clutter are various, going from planes to birds [26]. The algorithm responsible for the removal proves that each pixel has at least x neighbors with the same color —if it is not the case, the pixel is converted to a zero. The final value is a result of interpolation. Figure 2.4 uses $x = 16$ as the minimal amount of related pixels. Other methods for clutter removal include the texture-based algorithm developed in [14] which also uses a neighbour filter, generally with a wider window, and a test of compactness.

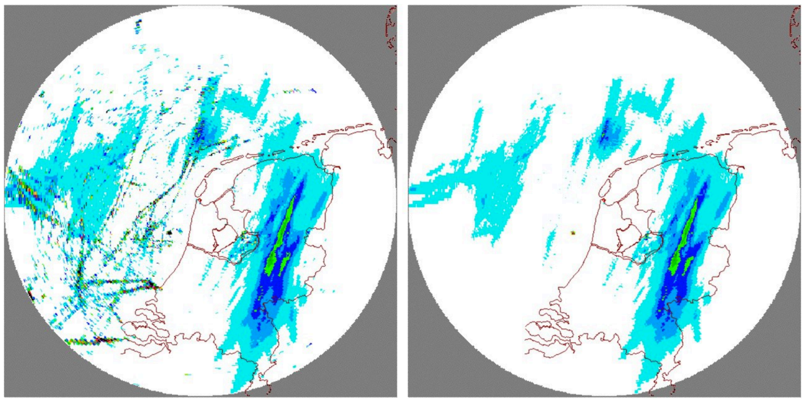


Figure 2.4: Radar image from the Royal Netherlands Meteorological Institute with and without clutter. Image extracted from [5].

Beam attenuation Attenuation happens when there are grid points with high levels of rain but low reflectivity values. It is usually caused when heavy rain debilitates the reflection of the beam on its way back to the radar. SCOUT tries to find intensity differences between the pixels of neighbor beams. It also uses the real rain gauge data to correct high discrepancies. In figure 2.5, the difference between the precipitation at the points marked with the cross and the reflectivity values was very high.

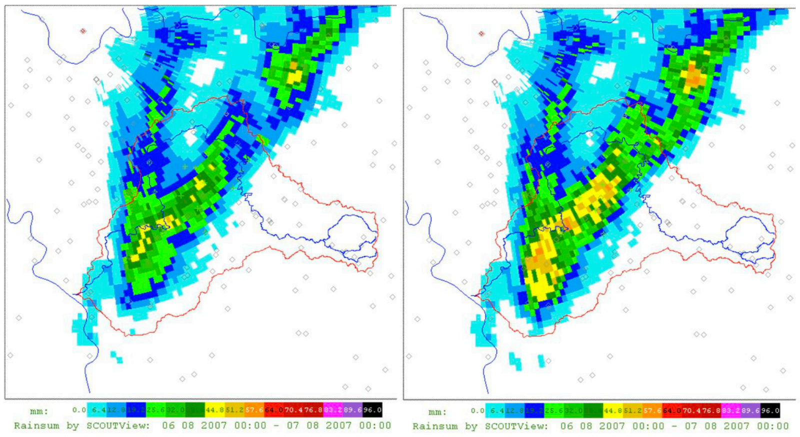

mm:
Rainsum by SCOUTView: 06 08 2007 00:00 - 07 08 2007 00:00
mm:
Rainsum by SCOUTView: 06 08 2007 00:00 - 07 08 2007 00:00

Figure 2.5: Radar image from Essen delivered by the German Weather Service before beam attenuation correction and after. Image extracted from [5].

Other quality control methods include adding filters to differentiate convective precipitation from the stratiform one —important for choosing the values of the Marshal & Palmer conversion—, the elimination of noise, or filtering of radar anomalies.

2.3.2 Image Analysis and Pattern Matching

In order to make a prediction of rain, it is necessary to know where the water of the clouds seen in the radar imagery is moving to. Information about wind direction alone is not a determinant factor because it can change by up to 180 degrees between relatively close points. That is why pattern-matching techniques are used. SCOUT marks clusters of convective cells and looks for those in the next picture. If found along several images, the accumulated spatial displacement shows the direction in trend. The image analysis and extrapolation technique of this method are better explained in [4] but it has been updated since the publication, especially regarding the tracking of cells, as is seen in [37].

The year 2011 saw the original cell-tracking algorithm being extended in five different ways. The innovation can be summarized in two techniques: the first one uses the semi-lagrangian scheme to calculate the displacement matrix. Every cell is searched for in the next prediction step and its growth (or degrowth) and new intensity are adjusted. The information that is later used to create the prediction is retrieved *backwards* for every cell, meaning that a future image is taken as a point of reference to find the patters of motion in the growth of the cell (it could be two consecutive images taken by the same radar). The second one uses the Lagrange Trajectory calculations to calculate the displacement matrix and makes a forward prediction instead of a backward one.

2.3.3 Conversion from reflectivity value Z to rain intensity R

After the state of the atmosphere for the analyzed radar images is predicted for the chosen time (max. future three hours), each pixel/Z value is converted to the rain intensity using the Marshal-Palmer relation with default values $a = 1.6$ and $b = 200$. As already mentioned, this conversion can result in high inaccuracies in the forecast.

The only possible way to offer a better prediction is to adjust a and b post-precipitation. This is an important service that SCOUT delivers for hydrology simulations. As mentioned in the introduction, it is not possible to measure the exact rainfall for every km^2 on open land. Such information would be especially relevant for drainage basins: the extensive area around any body of water that collects precipitation for said body. For example, a valley with a river in the middle constitutes the drainage basin for it. This means that heavy rainfall at several kilometers from the actual river can cause a rise in volume after a couple of weeks of even days. The tracking of water flow and volume in bodies of water is carried out thanks to hydrological simulations for which the values predicted —thanks to weather radar data images— are more reliable than the interpolation between rain recorders. SCOUT is able to use rainfall records to adjust the Z values of radar imagery and make better Z-R conversions [10]. Nevertheless, this is a computer-power hungry process that still contains inaccuracies and the SCOUT team welcomes and encourages further research in the short-term precipitation prediction field.

2.4 QPF via Artificial Neural Networks

Given the rise of machine learning algorithms and especially that of artificial neural networks in the last seven years, it was of interest for this thesis to research scientific applications of ANNs to weather forecasting in general and to QPFs in particular.

The few publications about ANNs applied to weather forecasting generally show successful results. Already in 1997, it was observed that possibility of precipitation (PoP) and QFPs could be determined by neural networks. The results were based on just 579 datasets and using up to 19 meteorological variables. The correlation between PoP and observed rain intensities reached 0.96, which was too promising and probably explained by the relative small amount of data [15]. Later, Cristofides (2000) tried using rainfall, wind speed and direction, temperature, humidity, and barometric pressure with simple ANNs to forecast PoP with discouraging results. The author however noted, that this was rather due to the lack of correlation between the used variables and precipitation and not due to the approach itself [2]. Culclasure (2013) achieved good results for temperature forecasting with lead times from three to 24h using ANNs [3], and Hoder (2013) managed better PoP than simple algorithms with forward ANNs using wind, humidity and barometric pressure variables[17]. Closer to our approach is Feng (2005), who used radar data and satellite estimated cloud-top temperature with ANNs to produce QPFs that generally outperformed linear regression [12]. The most relevant and best-performing work for this problem is the one done in [11], where complex ANNs were fed radar imagery to generate radar maps for the future one to six hours. The rainfall was then predicted using the Marshall-Palmer relation with $a = 118.239$ and $b = 1.5241$.

Chapter 3

Approach

This chapter explains the logical steps for choosing artificial neural networks as the statistical tool and briefly introduces ANNs in general and convolutional neural networks in particular.

3.1 Feature recognition on radar imagery using Convolutional Neural Networks.

A person looking at the video of weather radar data is able to detect clusters of cells with similar values/colors and its direction. Furthermore, a person trained to look at this kind of data will be able to intuitively say if there is going to be a lot or little rain. So, what if there is a mathematical formula that relates the amount of rain at a given point of time with the state of the atmosphere one hour before that? Since artificial neural networks (ANNs) are capable of approximating any function in $\mathbb{R}^n$ [25], if there is such a function, ANNs should be able to find it. The type of ANN chosen for this task is a convolutional neural network.

Convolutional Neural Networks (CNNs) have been used to classify images (e.g. [20], [7], [9]) with success. Moreover, CNNs have also been successful in regression problems. Bojarski *et al.* (2016) used CNNs to map images to steering angles [6]; this thesis will try to map radar images taken exactly one hour before the start of precipitation to its intensity. A further step will be to track the motion of the cells in the 1.5-hour video that can be put together from the given data. For that, this work will use the reknown optical flow algorithm of Lucas Kanade to follow certain pixels across the frames and determine the general direction of the clouds with the center of the images as a reference point.

3.1.1 Artificial Neural Networks

Artificial Neural Networks (ANN) are a biologically-inspired programming paradigm that enables a computer to learn from observational data and simulate any function in $\mathbb{R}^n$ [25]. The basic idea is that given enough inputs $x_{i_1}, ..., x_{i_n}$ that point to an output y_i, an ANN can approximate the function that relates the inputs and the output. It is also possible to have a multi-output function, but the focus is going to be on $\mathbb{R}^n \rightarrow \mathbb{R}$ functions because for every n amount of reflectivity values in the 101^2 km^2 grid, only the rainfall value for the center of that area is wanted.

The reasons for the increased use of ANNs as prediction systems are Big Data, the improvement of GPUs, and better algorithms. More efficient GPUs running more sophisticated algorithms allow for big complex networks that can find the myriad of different trends and patterns to be found in big data sets [24].

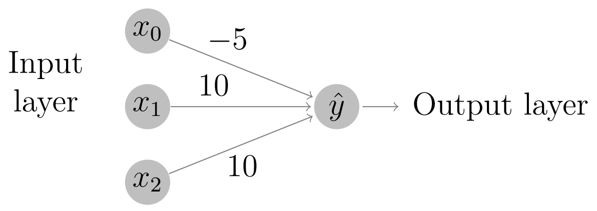
There are several components necessary to implement learning neural networks: an architecture, a training and a validation set, an activation function, a cost function, and hyperparameters.

The **architecture** is the input and output layer together with the number of hidden layers and nodes in each one plus the connections between each layer to the next one. The number of neurons in each layer represents its **depth**. A **training set** is formed by input values $x_1, ..., x_n$ that are mapped to a known result y. This set should be big enough for the inputs to contain significant variability. The **validation set** is a part of the training set upon which the network only does predictions that help assess the good of the training. The **activation function** is the one used to compute the values of each neuron, and the **cost function** is the one measuring the general accuracy of the network based on the error: the difference between each calculated value $\hat{y}$ and the real value y. The theoretical goal of the training is that the cost function reaches 0, meaning every prediction is perfect. A realistic approach tries to minimize it as much as possible while preventing overfitting: a challenge that will be discussed in the next section. The hyperparameters are the set of arguments that let the programmer experiment with an architecture to improve results —among this there is the number of batches, epochs, regularizers factor, dropout percentage, and the learning rate. Each of this terms will be briefly explained in the following sections as it appears.

Once the components are chosen, the learning has two fundamental steps: forward-propagation and backpropagation. The forward-propagation is the one carried out in the simple `OR` and `NOR` examples included below. This process computes the value of each neuron starting with the input values and initial (usually random) weights as feed and calculates an output value $\hat{y}$. The second type of propagation is the key to minimizing the cost function and was applied to multi-layer artificial neural networks by Geoffrey Hinton for the first time in 2006 [16]. His success led to the rise of ANNs as popular statistical tools for Machine Learning. Given a error $e = |\hat{y} - y|$, the backpropagation algorithm supposes that such error is $e' = e - learning\_rate$ and, using partial derivatives, the chain-rule, and dynamic programming, it calculates how the weights between the different layers have to change for the error to be e' and not e. After the weights are adjusted, forward-propagation takes place again. Each forward-backpropagation iteration is called an **epoch**, and the amount of such can be chosen by the implementer. The goal is that each epoch delivers a smaller error that eventually tends to zero.

The process of subtracting the learning rate from the error every time in order to find the minimum of the cost function is called **gradient descent**. It is a method to find the absolute minimum of the function, which is challenging due to the risk of overfitting.

In order to further explain the fundamental idea behind neural networks, the logical `OR` function will be simulated. The example network has one layer (the input one is not counted) and will have two inputs: x_1, x_2. Every layer also has a bias, x_0, which is always 1. The weights are $w_0 = -5, w_1 = 10, w_2 = 10$ and the activation function is the sigmoid one σ, which is shown in figure 3.1.1. Below is a visual representation of this network:



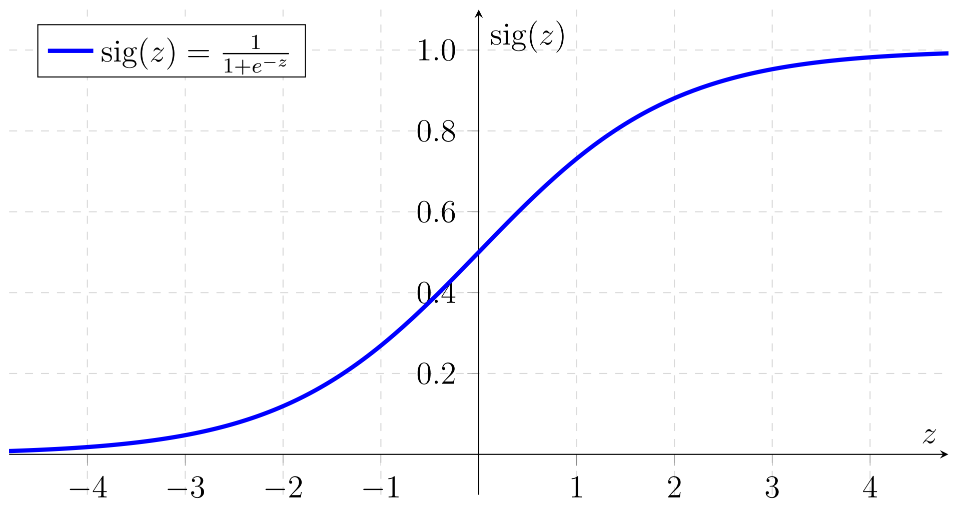


Figure 3.1: The sigmoid function smoothly goes from 0 to 1, with a middle point at 0.5. It is typically applied to logistic regressions for this reason, because the probabilities of rounding up or down to 1 or 0, respectively, are evenly distributed.

The value $\hat{y}$ is the result of the sigmoid function applied to a value z, which is the sum of every $x_i w_i$ result:

$$z = x_1 w_1 + x_2 w_2 + b$$
$$\hat{y} = \sigma(z)$$

In this case, the bias is $x_0 = 1$ times $w_0 = -5$, which is equivalent to say that $b = -5$. Replacing x_1 and x_2 with all possible combinations of 0 and 1 the result is the function `OR`:

x_1	x_2	z	$\sigma(z) \equiv p \vee q \equiv \hat{y}$
1	0	$1 \cdot 10 + 0 \cdot 10 - 5$	$\sigma(5) = 1$
1	1	$1 \cdot 10 + 1 \cdot 10 - 5$	$\sigma(15) = 1$
0	1	$0 \cdot 10 + 1 \cdot 10 - 5$	$\sigma(5) = 1$
0	0	$0 \cdot 10 + 0 \cdot 10 - 5$	$\sigma(-5) = 0$

Table 3.1: Logical function `OR` computed by an artificial neural network

The power of neural networks resides within the possibility of adding hidden layers and augmenting the amount of connections between neurons,

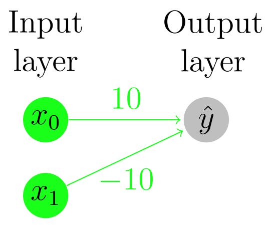


Figure 3.2: NOT function computed by an ANN

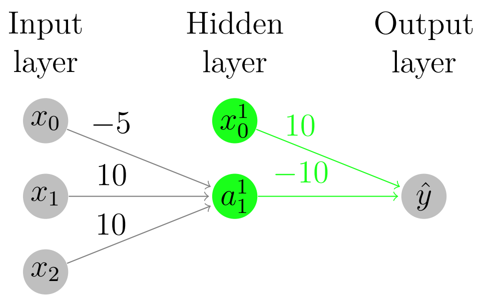


Figure 3.3: NOR gate computed by an ANN, where $a^1_{i,i\geq 1} = \sigma(z^1_i)$

very much like the human brain, which learns while neurons connect thanks to synapses. Wanting to showcase this feature, above is a brief example of the NOR operation, which is formed by connecting both the OR and the NOT gate. Figure 3.1.1 shows the neural network that computes the NOT function. This network was used to build NOR in figure 3.1.1, as proves table 3.2.

x_1	x_2	z_1^1	$\sigma(z_1^1) \equiv a_1^1$	z_1^2	$\sigma(z_1^2) \equiv \hat{y} \equiv NOR$
1	0	5	1	0	0
1	1	15	1	0	0
0	1	5	1	0	0
0	0	-5	0	10	1

Table 3.2: NOR gate computed with a composition of a neural network that computes OR and another one that computes NOT.

3.1.1.1 Challenges of Artificial Neural Networks

Overfitting. When a network is too big, it is possible that it starts to see the complex relations within the data as simple noise and stops learning to start *memorizing* the input data. This is called overfitting and is a typical problem to deal with when training neural networks. The result is a bad statistical model that overreacts to small changes in the data [35]. A sign of overfitting is when performance on the training data is much better than on the validation data. Solutions include adding more data, simplifying the network, stopping the learning at the first signs of overfitting, regularizers, and most recently, dropout:

- **Data Augmentation**. Data accumulation for Machine Learning purposes is generally expensive, so one can generate data synthetically. Data augmentation techniques which use images as input include creating new images by shifting the pixels horizontally and/or vertically, flipping —depending on the content— and reducing or increasing each pixel by the same amount [20].

- **Simpler Networks**. It is possible that there are too many hidden layers and these ones have too many neurons. This means that the complexity of the relation input/output is not so high and maybe simpler architectures could be more appropriate.

- **Early Stopping**. Another possibility is to stop the training as soon as the predictions on the training set keep getting better, but those on the validation set worse. The problem regarding this approach is that a function can have several minima, especially complex functions with more than two inputs. For example, in Figure 3.4, the function has a local minimum around $y = 1.6$. Were this the cost function, this minimum could have been reached and further learning would show higher values. Early stopping would then stop the training, although it is to be seen that keeping it would eventually deliver the absolute minimum. Prechelt concluded in [28] that being more patient produces better results in the long run, but also means significantly longer training times.

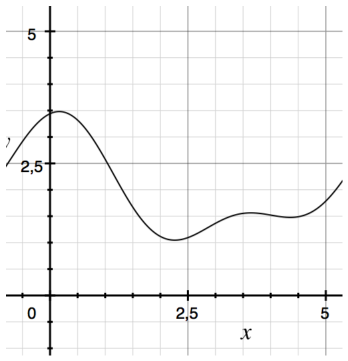


Figure 3.4: Function with a local minimum and an absolute one. Private image.

- **Regularization (weight decay)**. By adding a regularizing term to the cost function, the network is made to prefer smaller weights, which means that said network won't have big changes if the input values randomly changed. Big weights are only allowed when they represent a considerable improvement. It has been empirically proven that regularized networks generalize better [25].
- **Dropout**. This is also a type of regularization, but in this case, the whole network is modified. Dropout is performed by ignoring $p\%$ of the hidden neurons during the forward- and backpropagation. After adjusting the weights, a new epoch is run, this time ignoring a different $p\%$ of neurons. Heuristically, it is as if a a *different* network had been trained every time and the end model would be an average of all of them [35].
- **Stratify Data.** It can be that the distribution of the training set is very different from the distribution of the validation set or even not representative of reality. An exaggerated example would be training a temperature forecasting system with data from the Equator and using a validation set with data from the North Pole. It is therefore important to ensure all sets have similar distributions. If data is not stratified, either outliners can be ignored or misrepresented values can be synthetically augmented.

Underfitting. It happens when the network is not even learning from the training data. Causes can be architectures that are too simple or too few data with high intra-variance being used. The solution is generally a more complex network, better distributed data, and/or another statistical method.

3.1.2 Convolutional Neural Networks

Convolutional Neural Networks are Artificial Neural Networks that explicitly assume that the inputs are images, which allows the encoding of certain properties into the architecture. This vastly reduces the amount of parameters in the network [18]. ConvNets introduce new components such as filters/kernels and pooling layers and arrange neurons in three dimensions: width, height, and depth. The filters or kernels are matrices also with three dimensions that contain the weights. The dot product between the input's layer vector $(x_0, ..., x_n)$ and the first vector of weights $(w_0^1, ..., w_n^1)$ performed in simple neural networks is now the dot product of small parts of the image of the dimension of the filters and the filters themselves. The small parts are the results of dividing an image to squares of the same size and depth, just like in this figure:

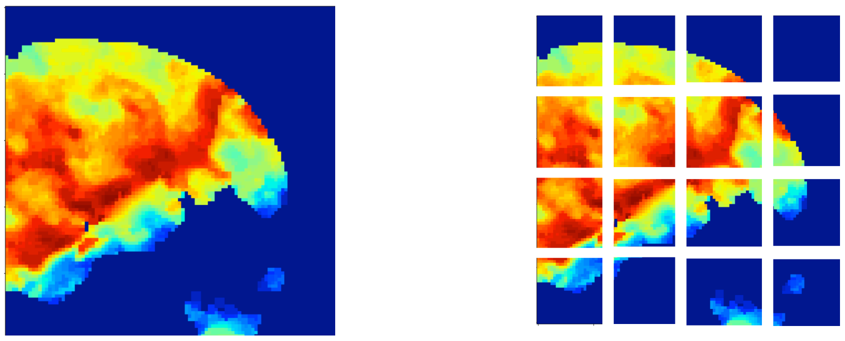
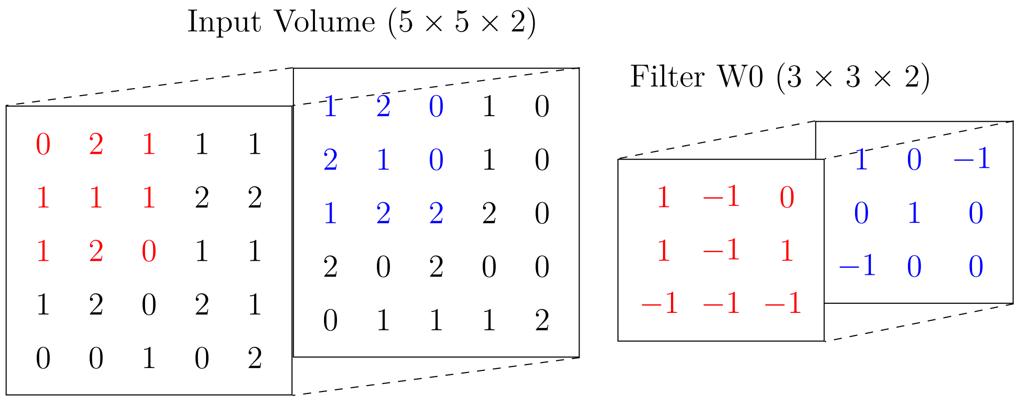


Figure 3.5: Left: image used to train ConvNet. Right: Division in sub-images that will be multiplied with the filters.

Each kernel concentrates on a different unknown feature and, given that an image can have thousands of different ones, another hyperparameter is needed: the depth of the output for each layer. This is the number of kernels for each epoch; the initial weights are different and, in the forward pass, each filter multiplies itself with all the sub-images to learn a certain feature

[18]. Thanks to the connections between the neurons, it is possible for the network to know that the order and amount in which the features are present is determinant to form one picture as opposed to another.

A brief example of the first step of a forward pass is presented below. It has an input image of dimensions $5 \times 5 \times 2$, one filter of size $3 \times 3 \times 2$, stride $S = 2$, and has been extracted and simplified from [18]. The stride S is the number of pixels that a kernel jumps when it slides to the next sub-image. In the example, the output volume is the value of the first neuron in the first hidden layer. Were there k filters, then it would be necessary to have k neurons in that layer. Also, no padding was used. The padding is used for cases in which the kernel would not have a subimage of the same size at the (horizontal or vertical) end of a picture because it is missing a row or a column. That missing vector within the sub-matrix can be padded with zeros and is called zero-padding.



The first value of the output volume is calculated by adding the dot product of the first 2D sub-matrix of the input volume with the size of the kernel and the kernel itself, plus the bias b, which is typically 1. This operation is carried out below. After the first value is calculated, the kernel is shifted to the right by S columns and calculates the second value. When done horizontally, it returns to the beginning of the matrix and skips S rows vertically.

$$\begin{pmatrix} 0 & 2 & 1 \\ 1 & 1 & 1 \\ 1 & 2 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & -1 & 0 \\ 1 & -1 & 1 \\ -1 & -1 & -1 \end{pmatrix} + \begin{pmatrix} 1 & 2 & 0 \\ 2 & 1 & 0 \\ 1 & 2 & 2 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix} + 1 = -2$$

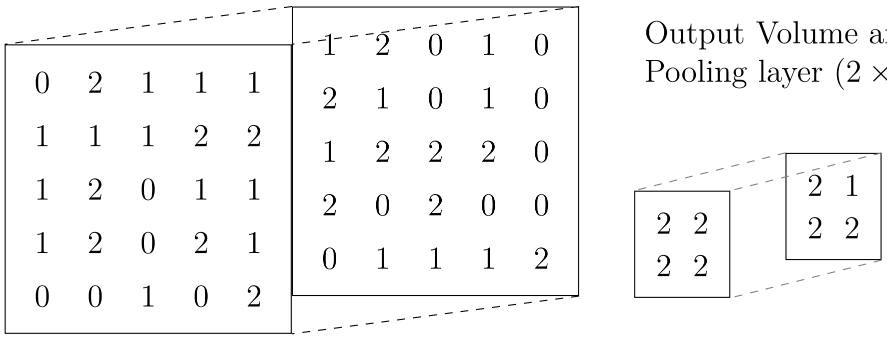
Output Volume $(2 \times 2 \times 1)$

$$\boxed{\begin{matrix} -2 & -1 \\ -3 & -3 \end{matrix}}$$

The other new component that ConvNets introduce is that of pooling layers, which are used to reduce the spatial size of the neurons and with it the overfitting. A pooling layer accepts an input with dimensions $W \times H \times D$ and uses a matrix smaller than the given shape to slide across the input and reduce each sub-matrix using a chosen function such as maximum or average. The input volume of figure 3.1.2 would look like the output volume in the following figure after applying a max-pooling layer with a matrix with the dimensions 2×2 and a stride of $S = 3$. As one can see, the spatial size can be considerably reduced whereas the depth stays untouched.

Input Volume $(5 \times 5 \times 2)$

Output Volume after Max-Pooling layer $(2 \times 2 \times 2)$



The success of convolutional neural networks is deeply connected to the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). It is a contest of high reputation based on the 1.2 million images of the ImageNet dabatase. In 2012, Krizhevsky, Sutskever, and Hinton achieved a top five test error of 15.4% (this metric is used to describe the rate at which a model does not classify an image correctly with its top five predictions). Considering that the second-best result was 26.2%, ConvNets proved a highly significant improvement in feature recognition and have been often used in similar problems since then, especially for the ILSVRC itself, where a team achieved a top five error rate of 6.7% in 2015 and won the contest [8]. For our approach, we will use some of the architectures used by the winners of the ILSVRC 2012, 2013 and by one of the top contestants of 2014. However, these models have been used for classification purposes, so it is appropriate to also use the Bojarksi *et al.* architecture, which was used to successfully regress pictures taken by a car to steering wheel angles. The best result out of one of these proven networks will be used to develop our forecasting model.

Chapter 4

Experimental Evaluation

Besides the given baseline of 14.69 mm/h given by the contest, we calculated the RMSE using the mean of the 10,000 rain values as prediction, which is 15.56 mm/h. The resulting RMSE is 15.85, leaving the product of the linear regression as the best mark to improve. The following chapters will first explore reknown architectures of ConvNets and will try to optimize the one that delivers the best results, as well as improve the input values using computer vision. However, before starting with the training of neural networks, we will show that both the training and the validation set are stratified and that there is no linear or simple polynomial relation between the input values.

4.1 Statistics of Data

Out of the given 10,000 observations, 80% will be the training set and 20% the validation set. We will keep the given order because it is chronological and attains to a realistic use case. With the histograms of the rain values of both sets, we can assert that the training set is representative enough for the validation set. As figure 4.1 shows, the distribution of the values is almost identical for both sets.

Once it was been shown that the data is stratified, it is important to examine how reflectivity values and rainfall values relate to each other in this scenario. One idea would be to take the average wind for mainland countries into account and use it as radius to crop around the center the images, but, if we look at the histograms of rain values, we can see that more than 40% of the datasets correspond to events of more than 10 mm/h rainfall, which surpasses the 7.6 mm/h at which precipitation is considered heavy [34]. Intense rain is a product of convective cells, which usually move very fast, so average winds should not be considered.

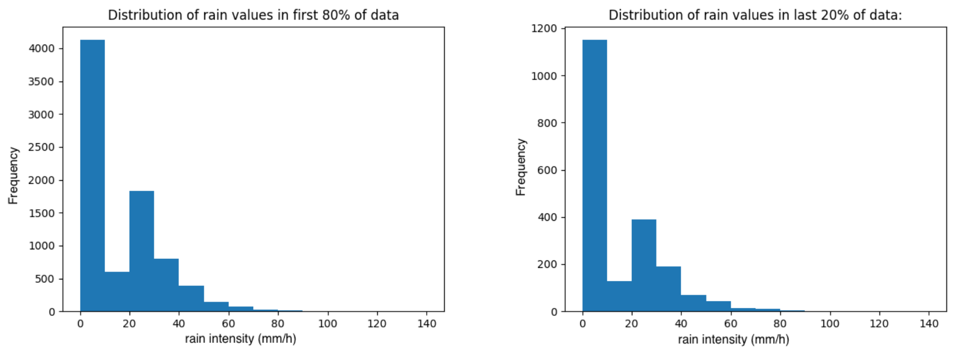


Figure 4.1: Histogram of training set's rain values (left) and histogram of validation set's rain values (right).

Also, the data has been scaled in an unknown way and there are reflectivity values higher than 70, whereas the regular maxima are 55-60 dBZ, that is why all Z values have been scaled to maximum 60 dBZ and later normalized. Figure 4.2 shows that there is no linear correlation between the mean Z values of the surrounding area and the precipitation value. This is also not expected, given that rain is naturally highly non linear due to its dependancy on the various atmosphere processes and their inherent chaos [27].

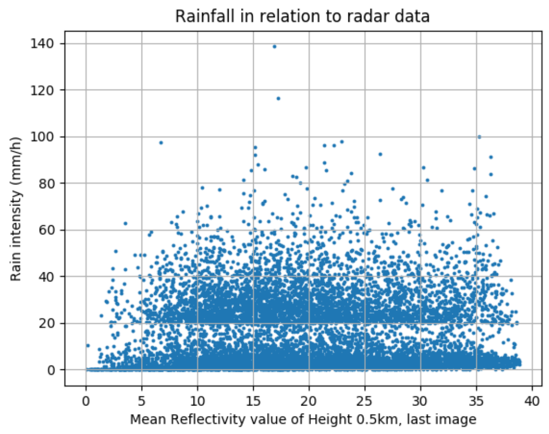


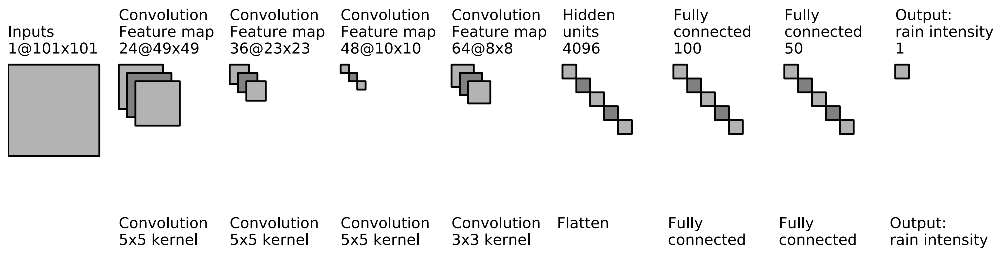
Figure 4.2: Mean Z value last image taken at 0.5 km mapped to rainfall at its center after one hour of this data being collected.

The non-linearity of rain is one of the reason to try neural networks to approach this problem. The next sections will document this attempt.

4.2 Handling weather radar as sequence of images

As seen in Chapter 3, Convolutional Neural Networks are designed to learn features from images and make decisions. Quantitative Precipitation Forecasting is a task that could benefit from this ability, so we will explore the results of using successful architectures to predict the future 1 to 2 hours rainfall. The only preprocessing at the stage is the clutter removal using the Gabella algorithm [14].

Bojarski et al. This team used convolutional neural networks to map images taken from a car to steering angles. Their architecture is shown in Figure 4.2 with the difference that the input size of their images is 66×200 and not 101×101. ConvNets trained using this architecture during 200 epochs, with a learning rate of 0.01, RMSE as cost function, and early stopping after 15 epochs have delivered the model loss shown in 4.2, where light overfitting starts around epoch 23.



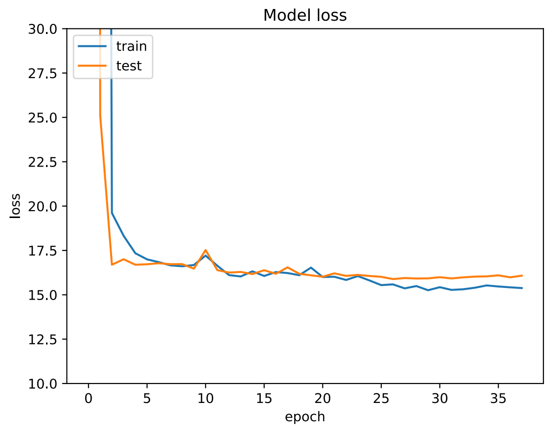
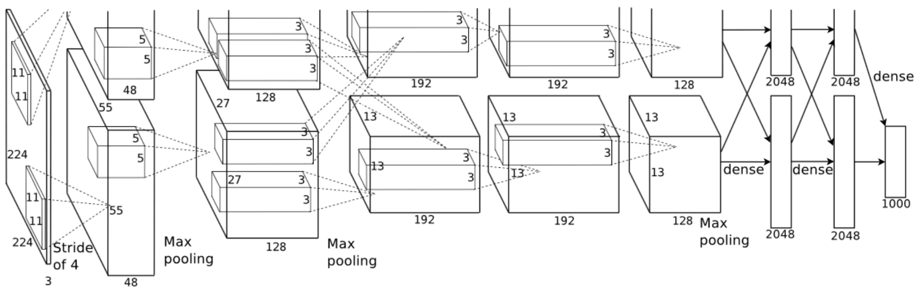


Figure 4.3: Above: Architecture of Bojarski et al. applied to a 101x101 radar image. Below: Model loss after training with Bojarski's architecture.

AlexNet. This is the name of the first ImageNet winning architecture in 2012. An illustration of the architecture that divides the load among two GPUs can be seen in figure 4.2. The input size of the ImageNet pictures was $224 \times 224 \times 3$, resulting on 200704 input values. Since we only have a 10th of that, 10201, we decremented the first kernel size from 11 to 7, have eliminated the fourth convLayer, and reduced to quantity of kernels from 96 to 48, from 256 to 128 and from 384 to 182, as well as the depth of the fully connected networks from 4096 to 512 [20]. The model loss can also be seen in 4.2.



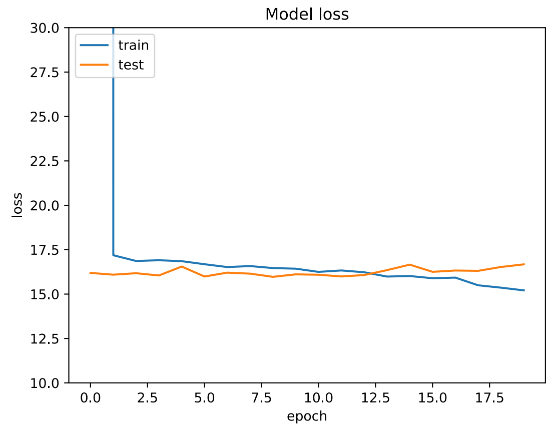
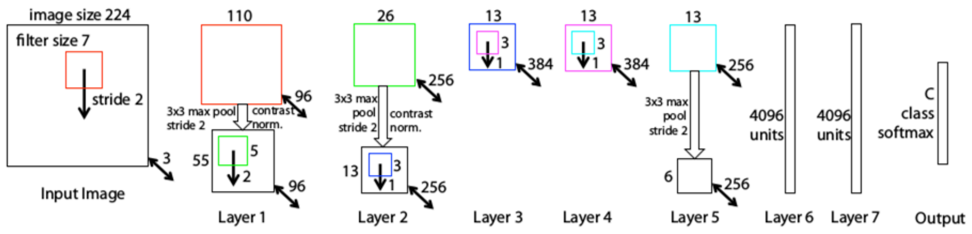


Figure 4.4: Above: First architecture to win the ILSVRC in 2012. Extracted from [20]. Below: Model loss after training with AlexNet architecture.

ZFNet. Very similar to the AlexNet, this network won the ImageNet Challenge in 2015 achieving an 11.2% error rate [39]. Its architecture and the model loss after training with it can be see in figure 4.2. Since the input images of the challenge are more than twice as bigger that ours, we decided to ignore layer 4 and 5 in order to avoid an overly complicated architecture.



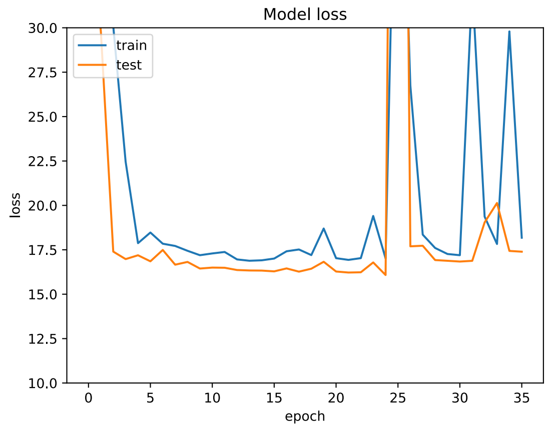
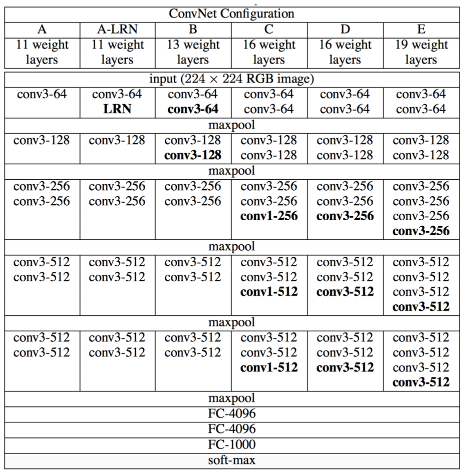


Figure 4.5: Above: Architecture similar to AlexNet developed by the ILSVRC winners of 2014. Extracted from [39]. Below: Model loss after training with ZFNet.

VGG Net. Even though the team of the University of Oxford did not win the challenge, their very simple but deep architecture achieved a 7.3% error rate [32]. Both the architecture and the model loss are in figure 4.2.

Aside from the models, it is possible to determine the good of the model by counting the amount of *hits* that each architecture hat. A correct guess in a regression is, unfortunately, not deterministic, that is why we have defined a *hit* as a prediction that is x mm/h from the real value. Values for x are 1, 3 and 5. Table 4.2 shows the number of hits, over predictions and underpredictions for each model.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

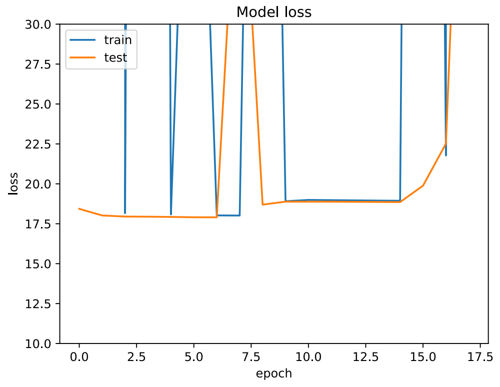


Figure 4.6: Left: Deep architecture developed by Symonyan and Zisserman. Extracted from [32]. The best results were achieved with option D. Right: Model loss after training with VGGNet D.

Architecture	x	hits	over predicted	under predicted	RMSE
Bojarski	1	**155**	844	1001	
Bojarski	3	**500**	613	887	16.21
Bojarski	5	**760**	401	839	
AlexNet	1	34	1215	751	
AlexNet	3	109	1171	720	16.46
AlexNet	5	220	1092	688	
ZFNet	1	**131**	963	906	
ZFNet	3	**437**	719	844	16.68
ZFNet	5	**697**	499	804	
VGGNet	1	20	1202	778	
VGGNet	3	73	1174	753	17.12
VGGNet	5	154	1114	732	

Table 4.1: Table showing the good of the predictions on the validation set by each chosen model. Bojarski and ZFNet seem to deliver the best results.

The predictions made by Bojarski's and the ZFNet architecture are considerably more accurate than the ones performed by AlexNet and VGGNet, however, ZFNet already contains dropout and regularization, so, from now and one, this thesis will focus on improving Bojarski's architecture, although the ZFNet one will still be trained with more data and with less deep hidden layers.

4.2.1 Reducing overfitting

The architectures has started to overfit and that is the reason it was stopped. As seen in Chapter 3, a probable reason is that it is too complex for the relative small amount of data that we have, especially if we consider that its original inputs were at least twice as a big as ours. That is why the first approach to reduce overfitting will be to augment the data by flipping the images on their vertical axis and using the same architecture.

Data augmentation. The only possibility to augment the given data is to flip it using any of the two axes. Shifting the pixels in a horizontal or vertical way is not recommendable because there are datasets in which convective cells only start to appear on the sides, but already with high dBZ values that generally mean strong winds and therefore a high chance of affecting the precipitation intensity at the center of the image. When flipping, special attention has been given to keeping each flipped image after the original one in order to keep the chronological order. The results of both architectures trained with augmented data can be seen in 4.2.1. Bojarski's model presents a highly irregular learning on the validation set, which is interpreted as high overfitting. This is a contrast to the mild overfitting shown in Figure 4.2 (below).

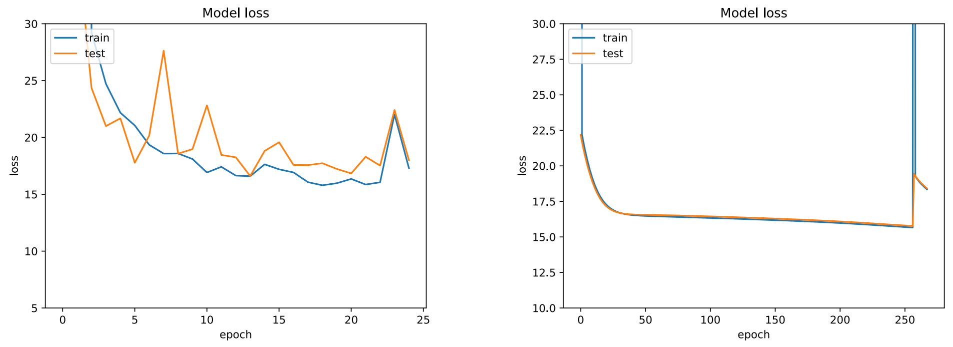


Figure 4.7: Models after training with augmented data. The left one is Bojarski's, the right one the ZFNet.

Simple networks. The original input image for Bojarski's architecture hat dimensions $66 \times 200 \times 3$, meaning there were almost 4 times more input neurons than the ones this thesis is using ($101 \times 101 = 10201$). Moreover, ZFNet was created to be trained with images with dimensions $224 \times 224 \times 3 = 150128$ neurons, about 15 times more that the ones delivered by the weather radar. It is therefore possible, that the overfitting is a result of too many and/or too deep hidden layers, so the first step to simplify the networks will be to reduce the depth of each layer in Bojarski's architecture and ZFNet by 2. Figure 4.2.1 shows the results of the simpler architectures trained first with the original dataset, and then with augmented data. Whereas Bojarski's architecture learns and overfits, the ZFNet one stagnates at a fixed value, meaning the network is unable to learn from the given data.

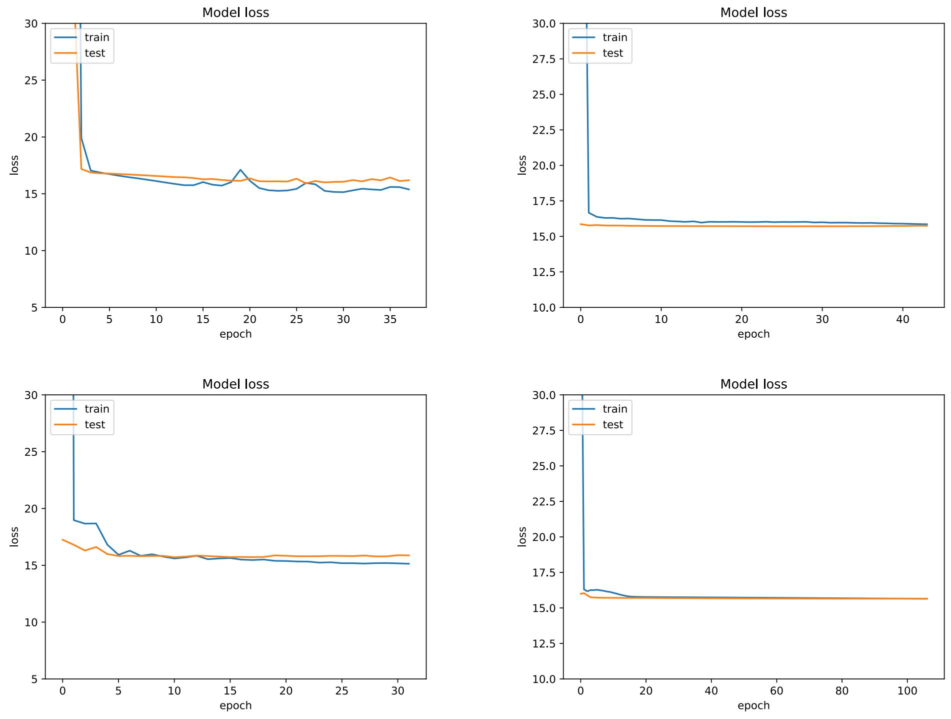


Figure 4.8: Left: Bojarski's architecture. Right: ZFNet architecture. Result of simplified networks trained with augmented data (models below) and without (models above).

Dropout. The ZFNet architecture already uses the a 50% dropout after the first fully-connected layer. This percentage is the one recommended by the paper introducing this technique [35], so it is the one to be used in Bojarski's architecture, once with the original architecture, then with augmented data, and then simplified.

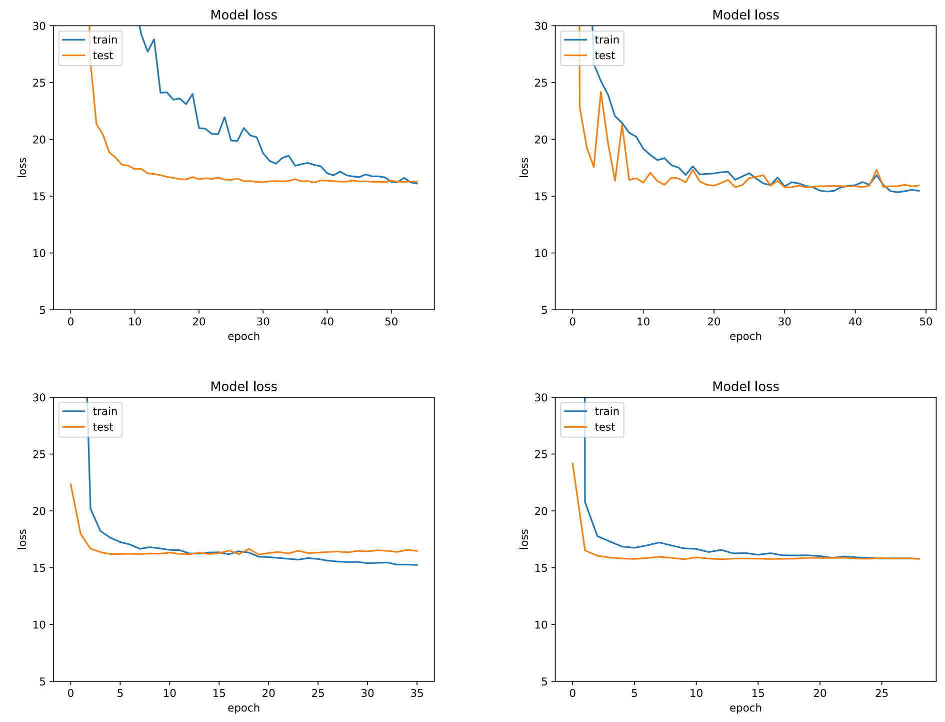


Figure 4.9: Models showing the learning of the Bojarski's architecture with a 50% dropout. The first one (order left-right, up-down) was trained with the original architecture, the second one with added synthetic data, and the third one with a simplified architecture. The last one is the resulting model after applying all techniques and had the best performance with an RMSE of 15.67 mm/h.

The first model using the original architecture and dropout shows very well how this technique reduces overfitting. Whereas the forward-propagation is done using just a 50% of the network, the predictions on the validation set are carried out using the whole network, causing that the validation error is lower than the training error.

Regularizers. The last resource to try to reduce overfitting is to add a regularizing term to the cost function. The team of the AlexNet architecture used a weight decay of 0.0005 [20] for their training with very good results, so that will be the value used. Figure 4.2.1 shows the model as a result of training with a simplified architecture, 50% dropout, augmented data, and regularization.

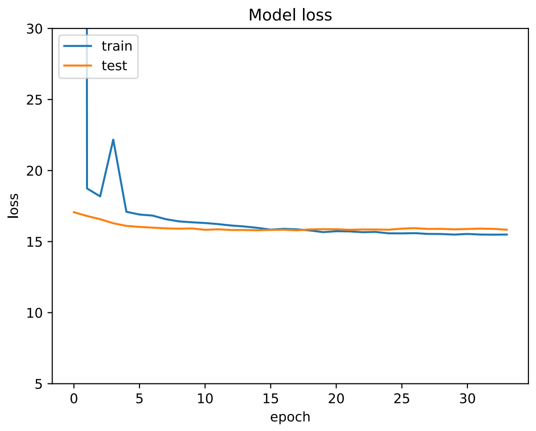


Figure 4.10: Model trained using all anti-overfitting techniques. The result was a RMSE of 15.79 mm/h, so no better than without regularization, but with a smoother learning.

No technique to reduce overfitting has cause a considerable improvement in the learning of the neural network. The reason for this is that the network may be learning too many irrelevant features for the final prediction. Searched for is the area that will have a higher chance of being over the km^2 center of the image within one hour, meaning that not all 101×101 pixels are important and are therefore introduce noise to the model. The next task will be to determine which part of the each training image can be cropped.

4.3 Finding motion with Lucas Kanade

The current results would probably improve if we could only consider the part of the clouds that are moving towards the center of the images. Even though wind directions can be opposite between two relatively close points, it is worth trying to determine the general motion of the grid in order to discard part of it for the training. Optical Flow is a field of research within Computer Vision and offers the Lucas Kanade algorithm as a way to estimate motion in videos. Using the consecutive 15 images at the lowest height, we have created a video for each dataset.

The Lucas Kanade method chooses interesting points in the first frame and tracks them in the next ones, making it possible to draw the vector movements of each chosen pixel across the video [22]. The procedure to algorithmically discern the direction using this lines is assembled as this: the Lucas Kanade (LK) algorithm delivers a set of old coordinates and one of new coordinates after each iteration. For each coordinate, we have calculated if the movement was in direction NE, NW, SE, or SW. Four counters have kept track of the amount of movements in each direction. Once LK is done, the chosen direction is the majority one. If, for example, the amount between NE and NW movements is close, the direction of motion determined is just North. An example is shown in Figure 4.11, where the mentioned method correctly chose 'Northeast'.

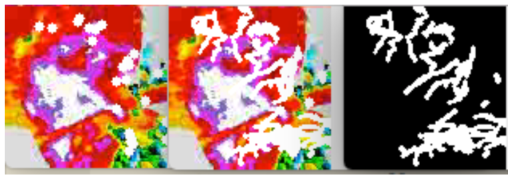


Figure 4.11: Left: Frame with selected pixels to track (white points). Center: Frame after the Lucas Kanade method draws lines showing every movement of every chosen pixel. Right: Only the drawn lines.

Once all directions are determined, only a 50×50 km^2 area is cropped from each original grid based on each direction. For the above example with the clouds generally going to the NE, only the low left part of the grid was considered for the training. Figure 4.12 shows the result of training the simplified version of Bojarski's architecture with augmented data, and a 50% dropout.

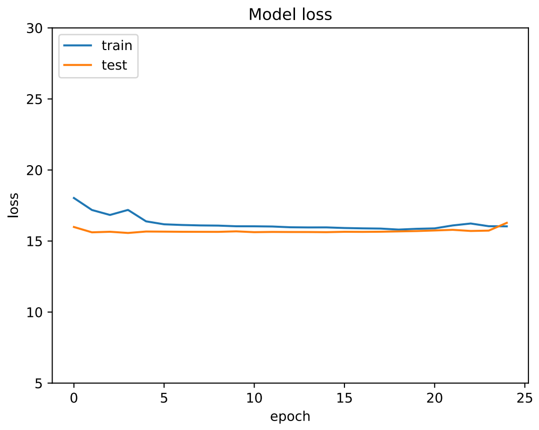


Figure 4.12: This architecture reached an RMSE of 15.53 mm/h, lower than any of the models trained with the complete images.

Although the latter results are better than those of networks trained with the complete images, it was observed that a considerable number of directions were wrong, which is why an alternative to determine motion was searched for.

4.3.1 Determining the points to track using SIFT features

The default Lucas Kanade implementation that the openCV library has in its tutorial uses the function `cv2.goodFeaturesToTrack`, which implements the Shi-Tomasi Corner Detector algorithm to find the set of pixels to start tracking [38]. Given that some datasets have very few clouds to be seen, it is interesting to try an algorithm that finds features all over the picture and not just at the corners. The Scale Invariant Feature Transform (SIFT) algorithm was introduced by Lowe in 1999 and has been very successful in feature matching approaches. It first creates a scale space of images, then finds differences of each pair of images and local extrema in the scale space, which helps locate the keypoints [21]. It has been described as one of the best feature-matching algorithms by Mikolajczyk & Schmid (2005). When used to find points to track in the radar weather images, it consistently found a considerable higher amount than the Shi-Tomasi implementation of the openCV library. Figure 4.3.1 shows an example of both algorithms applied to the same frame, where the differences in number of features is clear.

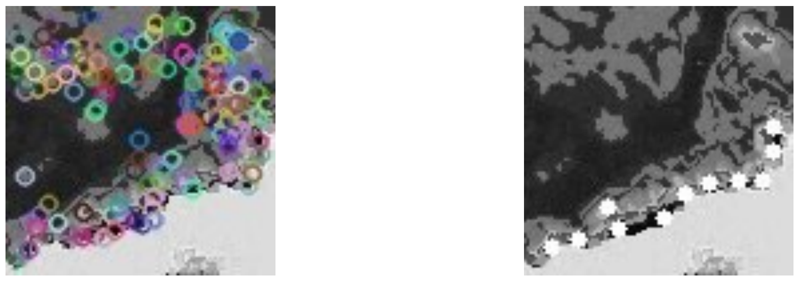


Figure 4.13: Left: Point to track found by SIFT algorithm. Right: Points to track found by Shi-Tomasi algorithm.

Figure 4.14 shows the results of training the same architecture that reached an RMSE of 15.53 mm/h with images that have been cropped using the directions determined by using the SIFT algorithm to extract points to track and not the Shi-Tomasi one.

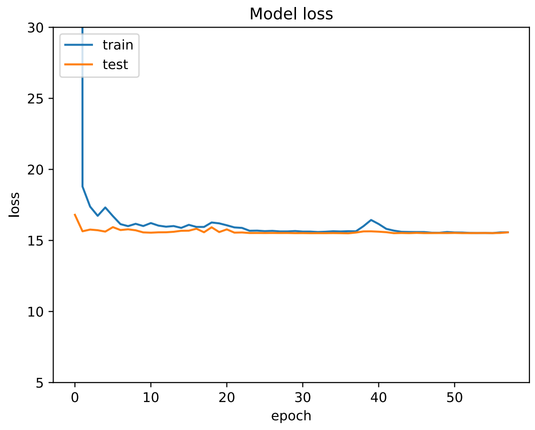


Figure 4.14: The lowest RMSE reached by this model is 15.51 mm/h.

4.4 Comparison of SCOUT and best performing Convolutional Neural Network

A scientific thourough comparison should be performed upon predictions on the same dataset, or at least similar ones. It is therefore important to stress that this comparison lacks the desired scientific rigor. The reason for this is that SCOUT and the best performing CNN developed in this work could not use the same dataset to make predictions. The CIKM dataset lacked information necessary for the Data Quality Control that SCOUT carries out, explained in Chapter 2, and SCOUT's dataset was in a format that demanded a high programming effort to be converted to a training set usable by the CNN used in this thesis, such that programming that conversion was not possible due to time constraints. Figure 4.4 serves as a proof of the disparity of distribution of precipitation intensities in both datasets.
Since the CIKM AnalytiCup contest 2017 has not disclosed any information about region, season, or general climatology, the predictions compared could be based on complete opposite scenarios. In order to reduce the probability of this happening, only Summer predictions of Scout have been taken into account. The logic behind this decision is that there are 5173 observations from the CIKM dataset with a rainfall value higher than 7 mm/h, which means that more than 50% of the events had heavy rain [34]. Intense precipitation over the timespan of one hour is a typical sign of thunderstorms (convective cells) —a summertime phenomenon—, which implies that, either the data was collected in a geographical area close to the Tropics, or it was during Summer. Actually, two facts point to the dataset belonging to a tropical or subtropical geographical area: first, the weather imagery is provided by the Shenzen Meteorological Buearu, which is on the border of the Tropic of Cancer, and second, the maximum observation in the dataset is 138.4 mm/h, a value more typical of the Tropics. As a contrast, the maximum rainfall value within the data used for SCOUT —corresponding to northern Germany— is 35.5 mm/h. A dataset from a tropical location was searched for to carry out the comparison, but weather imagery tends to be proprietary.

Nevertheless, the comparison can serve as a small pointer of the good of the model developed in this thesis, as well as an indicator to determine if it is recommended to pursue further research of Artificial Neural Networks as short-term Quantitative Precipitation Forecasting systems.

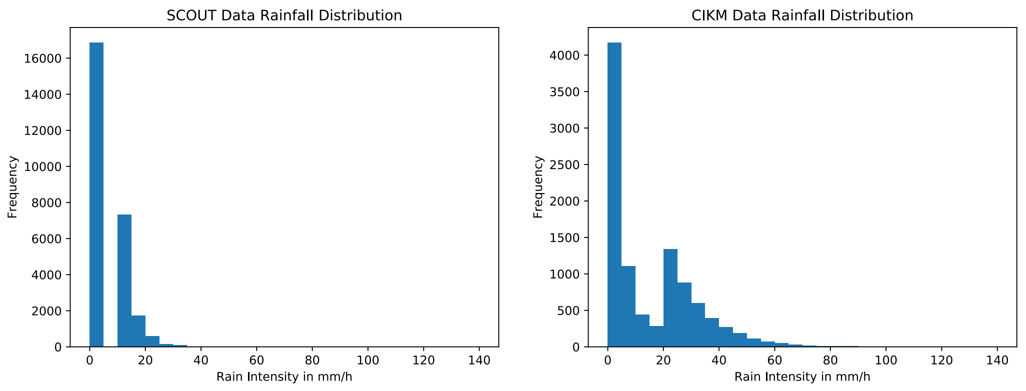


Figure 4.15: The distribution of both datasets is very different. The CIKM data has a determinant higher variance with a number outliers that make statistical learning more difficult.

The dataset used to make predictions with SCOUT corresponds to hourly measurements collected in the North of Germany during the Summer of 2015 (exact dates are June 11th to September 11th). The area covered 233 rain gauges and the RMSE was calculated based on the average precipitation recorded at these weather stations and the average rainfall forecast for all rain recorders. Also, since the CIKM dataset only has a 4% of rainless events, only non-zero rainfall values have been taken into consideration for the calculation. The result is an RMSE of 13.98 mm/h against the best result delivered by the Bojarski's architecture, 15.51 mm/h. However, if we consider the fact that the highest rainfall value in SCOUT's dataset is 35.5 mm/h and that there are more than 200 observations over that value in the CIKM's validation set, the result is significantly different. When predicting precipitation intensity for events with less than 35.5 mm/h, the resulting RMSE is 11.20 mm/h. Table 4.4 shows a comparative overview —the architecture in this table is a result of simplifying Bojarski's architecture, adding 50% dropout to it, and training with imagery cropped with directions determined by SIFT and Lucas Kanade, as well as augmented data.

QPF System	RMSE (mm/h)	max. rainfall value (mm/h)
Bojarski's architecture (modified)	15.51	95.4
Bojarski architecutre (modified)	11.20	35.5
SCOUT	13.98	35.5

Table 4.2: Comparative of RMSE metric reached by SCOUT and this thesis' best performing CNN.

Looking at the table, a strong valid affirmation on the good of the developed architecture is not possible. Were the performance of SCOUT on datasets with a significant number of outliers and high variance also close to an RMSE of 15%, it would be a strong argument to indicate that further research on ANNs as QPFs should be performed.

4.5 Discussion

This chapter started by mentioning two baselines for the RMSE: 15.56 and 14.69 mm/h. The first one is the result of using the mean of the rainfall values as prediction, and the second one a result of linear regression (LR) carried out by the contest organizers. Being rain a highly non-linear phenomenon, it is surprising that a CNN had a worse performance than a LR and only a slight better one that the baseline reached with the mean. Considering that the winner of the CIKM AnalytiCup 2017 also used CNNs —disclosed in private communication by organizers— to reach an RMSE of 10.99 mm/h, it can be strongly suggested that one of the reasons for the bad performance of the CNNs of this thesis is a failure in the extraction of meaningful features for the training. It should be stressed that the radar imagery for each rainfall observation included both horizontal and vertical information, as explained in Chapter 2. This thesis used the horizontal one to extract the direction of motion, but not the vertical information. This decision was made after private consulting with researchers of weather forecasting. They pointed to the fact that the precipitation to fall within one hour was most likely to be influenced by the atmosphere at 0.5 km and not at 1.5 km or higher. Other possible source of valuable features could have been to use the locations of tracked clusters of convective cells as extra training data.

Chapter 5

Conclusion

The goal of this thesis was to develop a short-term quantitative precipitation forecasting system using Artificial Neural Networks as a training model.

5.1 Summary

Chapter 1 outlines the motivation for these type of systems: activate protocols to avoid both human and material losses in case in imminent heavy precipitation and improve hydrological simulations that can help predict and prevent floods with days to weeks of head time. It also explains that this task was first presented by the CIKM AnalytiCup 2017 challenge, which set a RMSE of 14.69 mm/h as a baseline. The first three contestants out of the 1395 teams managed an RMSE of 10.99, 12.33, and 12.95 mm/h, respectively. Only 85 teams improved the given baseline. A model developed during the writing of this thesis ended in position 68[1] and the best performing CNN of this work did not beat the contest's baseline.
Chapter 2 introduces relevant background information about weather radars and precipitation needed to understand the data and to extract meaningful clues. It also summarizes the different state-of-the-art QPF systems used by the international community. These are characterised by Ensemble Prediction Systems (EPS), which are the results of simulating several diverse

[1]The RMSE that reached this position, 14.30581 mm/h, was achieved by a strongly overfitted Simple Recurrent Neural Network (RNN) trained with randomized values. Further trainings with RNNs neither reproduced that value, nor improved the metrics achieved with CNNs, so the focus of the research remained on the later type of networks.

scenarios in the atmosphere. An example of a low resolution EPS to be used for short-term predictions is the COSMO-DE model, used by the German Weather Service. Unfortunately, the verification reports obtained for this model do not include any continuous metric on quantitative precipitation forecasting. Actually, COSMO-DE only predicts the probability of rainfall exceeding 5 and 10 mm/h [31]. SCOUT, the system used for the comparison against our model, was also introduced in this chapter. It outputs hourly rainfall forecasts using pattern-matching solutions to follow convective cells and then the Marshall-Palmer relation to translate the dBZ values of the tracked cells into rainfall. It uses weather imagery collected by COSMO-DE, among others. Finally, a concise outline of uses of Artificial Neural Networks for weather forecasting was done, showing that there are not many recent examples of such approaches.

Chapter 3 explains the reasoning behind using ANNs and CNNs specifically as the approach for the task. A brief general explanation of neural networks followed with the goal of showcasing the necessary components, the general idea behind this mathematical paradigms, and the reason for their resurgence and recent successes, as well as typical challenges that they present and ways to tackle those challenges.

Chapter 4 includes the results of using known successful neural network architectures to train models with 8000 observations and validate with 2000. The first round showed that the Bojarski's architecture and the ZFNet delivered the best results, but still not better than the contest's baseline. They both had clear signs of overfitting, so a list of measurements applied to help the networks better generalize was documented. Since the results did not improve, it was determined to be appropriate to crop the images according to the motion detected and videos of the weather radar imagery were developed for this purpose. With these videos and features extracted first with the Shi-Tomasi algorithm and then with the SIFT one, we used the Lucas Kanade algorithm and simple logic to determine the direction of the motion. Cropping the images delivered the best result (15.51 mm/h). Finally, this result was compared with the RMSE of SCOUT on three months of data.

In reference to the thesis goals, it can be stated that the RMSE baseline set by the CIKM contest was not improved by the best performing system developed in this thesis. However, regarding the second goal, the results can be interpreted in a positive way. Although the RMSE calculated based on the CIKM dataset is worse than that reached by the SCOUT system, when comparing the quality criteria of both systems applied to more similarly distributed datasets, the CNN performs better than SCOUT.

5.2 Challenges and Future Work

The biggest obstacle of the challenge presented by this thesis has been the wide spectrum of the application domain. A completely rigorous approach would include deep knowledge of physics and mathematics (fluid mechanics, chaos theory, Lagrange schemes), meteorology (forming of precipitation), computer vision (optical flow), pattern-matching algorithms, and Machine Learning techniques. It was difficult to determine how much research was needed in every mentioned topic in order to make adequate progress.

Another challenge has been the lack of information about the CIKM dataset. Weather radar data can be best preprocessed when in polar form, but only the Cartesian format was made available. The conversion from one format into the other is possible, but additional information is necessary. Also, not knowing about the geographical position of the location of the radar affected the quality of the comparison to SCOUT, as discussed in Chapter 4. One attempt to overcome this challenge was to look for third-party weather imagery, but this one is proprietary and acquiring research rights for it was not possible. Moreover, the performance reports released by meteorological institutions such as German Weather Service and even the ECMWF do not include continuous metrics for QPFs.

Regarding future work, there is plenty research to be accomplished. First of all, it is necessary to train with much more data than 10,000 observations. In comparison, the architectures used in Chapter 4 were developed using more than 2 million. Looking at the right histogram of rainfall in Figure 4.4, it is possible to see that the observations between 5 and 20 mm/h are too few for a network to be able to securely predict such values. A dataset closer to a normal distribution could have very positive results.

Another future improvement could be using Recurrent Convolutional Neural Networks. Recurrent networks are a type of ANNs capable of making predictions taking facts from past step into account —they have *memory.* It is mentioned in Chapter 2 that Feng [11] already uses this type of network to predict future radar images with high success. Finally, the Marshall-Palmer relation could be revisited using neural networks. Even if the Recurrent Neural Networks or Ensemble Prediction Systems manage to generate the next images very accurately, the handicap of predicting rainfall seems to be the mentioned relation because using fixed values for it can result in big forecasting errors, even if the predicted dBZ was correct. These values are calculated post-precipitation based on the drop-size distribution. Given that drop-size distributions depend on the height and the development of clouds, further research using information about the vertical profile of the atmosphere — available in the CIKM dataset— would be valuable. A continuation of this research using Machine Learning techniques to map information on height and time to the values a and b of the Marshall-Palmer relation is one possible direction. Another avenue is to forget about this relation and try to train CNNs with small areas around the dBZ values (and probably also the same area at different heights) to look for a more accurate conversion of those to precipitation values.

References

[1] Shenzhen Meteorological Bureau. Cikm analyticup 2017: Short-term quantitative precipiattion forecasting, 2017. URL: https://tianchi.aliyun.com/competition/information.htm?spm=5176.100068.5678.2.63a4acfeaop0Xk&raceId=231596 (visited on September 20th, 2017).

[2] A. Christofides. Short-term rain prediction with artifical neural networks. Master's thesis, University of Manchester, 2000.

[3] A. Culclasure. Using neural networks to provide local weather forecasts. Master's thesis, Georgia Southern University, Electronic Theses & Dissertations, 2013.

[4] T. Einfalt. *Recherche d'une methode optimale de prevision de pluie par radar en hydrologie urbaine*. PhD thesis, Ecole Nationale des Ponts et Chaussées, 1988.

[5] T. Einfalt and C. Fenning. Scout dokumentation, 2011. hydro & meteo GmbH&Co. KG.

[6] Bojarski et al. Explaining how a deep neural network trained with end-to-end learning steers a car. *arXiv preprint arXiv:1604.07316*, 2016.

[7] Karpathy et al. Large-scale video classification with convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[8] Olga Russakovsky et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[9] Oquab et al. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

[10] Thorndahl et al. Weather radar rainfall data in urban hydrology. *Hydrology and Earth System Sciences*, 21(3):1359, 2017.

[11] Xingjian et al. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.

[12] Y. Feng and D. Kitzmiller. A short-range quantitative precipitation forecast algorithm using back-propagation neural network approach. *Advances in Atmospheric Sciences*, 23(3):405–414, 2006.

[13] European Center for Medium-Range Weather Forecasts. Who are we, 2017. URL: https://www.ecmwf.int/en/about/who-we-are (visited on July 5th, 2017).

[14] M. Gabella and R. Notarpietro. Ground clutter characterization and elimination in mountainous terrain. In *Proceedings of ERAD*, volume 305, 2002.

[15] T. Hall, H. Brooks, and C. Doswell. Precipitation forecasting using a neural network. *Weather and forecasting*, 14(3):338–345, 1999.

[16] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[17] C. Hoder and B. Southworth. Forecasting precipitation in hanover, nh using neural networks. *Dartmouth Undergraduate Journal of Science*, 05 2013. URL: http://dujs.dartmouth.edu/2013/05/forecasting-precipitation-in-hanover-nh-using-neural-networks/#.WWja1IqxV25 (visited on July 10th, 2017).

[18] A. Karphaty. Convolutional Neural Networks for Visual Recognition Course, 2017. Faculty of Computer Sciences, Stanford University, URL:http://cs231n.github.io/convolutional-networks/ (visited on July 10th, 2017).

[19] S. Klaus and C. Schraff. Improvements of the operational latent heat nudging scheme used in cosmo-de at dwd. *COSMO Newsletter*, 9(7):11, 2008.

[20] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[21] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[22] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[23] J. Marshall and W. Palmer. The distribution of raindrops with size. *Journal of Meteorology*, 5(4):165–166, 1948.

[24] A. Ng. Why is deep learning taking off. URL: https://www.coursera.org/learn/neural-networks-deep-learning/lecture/praGm/why-is-deep-learning-taking-off (visited on August 25th, 2017), 2017.

[25] M. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. URL: http://neuralnetworksanddeeplearning.com/ (visited on July 1st, 2017.

[26] Association of German Engineers (VDI). *Environmental meteorology: Gound-based remote sensing of precipitation, Weather radar*. Beuth Verlag GmbH, 1 edition, 2013.

[27] World Meteorological Organization. Guidelines on ensemble prediction, 2012.

[28] Lutz Prechelt. Early stopping-but when? *Neural Networks: Tricks of the trade*, pages 553–553, 1998.

[29] S. Pulkkinen. Identification and nowcasting of convective storms. Finnish Meteorological Institute, ERICHA, 2017. URL: http://ericha.eu/training/resources/pdf/Seppo_Pulkkinen_Convective_Systems.pdf (visited on June 25th, 2017).

[30] T. Schumman. 20170630 - heavy rain over ne part of germany, especially in the greater berlin area, 2017.

[31] German Weather Service. Verifikationsbericht zur güte lokaler wettervorhersagen und warnungen. auswertung zum winterhalbjahr 2016/2017. *Geschäftsbereich Wettervorhersage*, (56):76, 2017.

[32] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[33] American Meteorological Society. *Glossary of Meteorology*. Cambridge University Press, 2012. URL: http://glossary.ametsoc.org/wiki/Runoff (visited on August 12th, 2017).

[34] American Meteorological Society. *Glossary of Meteorology*. Cambridge University Press, 2012. URL: http://glossary.ametsoc.org/wiki/Rain (visited on September 12th, 2017).

[35] Hinton G. Srivastava, N. et al. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[36] K. Stephan, S. Klink, and C. Schraff. Assimilation of radar-derived rain rates into the convective-scale model cosmo-de at dwd. *Quarterly Journal of the Royal Meteorological Society*, 134(634):1315–1326, 2008.

[37] A. Tessendorf and T. Einfalt. Ensemble radar nowcasts a multi-method approach. *IAHS-AISH publication*, pages 311–316, 2012.

[38] Open Source Computer Vision. Optical flow, 12 2013. URL: https://docs.opencv.org/3.2.0/d7/d8b/tutorial_py_lucas_kanade.html (visited on August 5th, 2017).

[39] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.