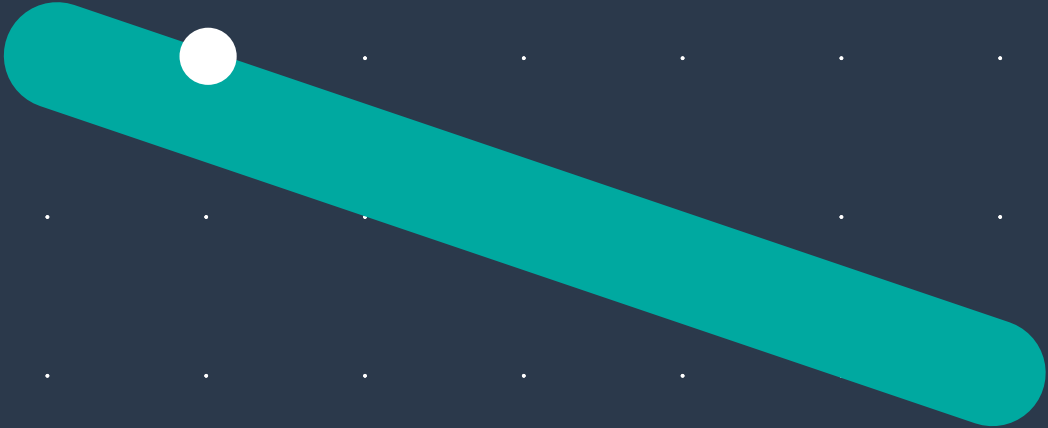


Submitted by
Simon Christofzik
Paul Sutter
Till Reitlinger



Teamproject (Master 3. semester)

DeepRain: Rain forecast with neural networks and the visualization of these in an App

Extended Abstract

Topic:	DeepRain: Rain forecast with neural networks and the visualization of these in an App
Team members:	Simon Christofzik, Paul Sutter, Till Reitlinger
Advisor:	HTWG Konstanz - University of Applied Sciences HTWG Konstanz - Institute for Optical Systems Prof. Dr. Oliver Dürr

The goal of the present work is to examine whether it is possible to calculate a rainfall forecast with limited resources and to make it available to users. For the calculation of the rain forecast neural networks were used. The required historical and current radar data were obtained from the German Weather Service and then analyzed and processed. Furthermore, an app was developed in which the rain forecasts are visualized. It also offers the possibility to notify the user in case of imminent rain. All the code and the full length documentation can be found on GitHub: <https://github.com/PaulIVI/DeepRain2>.

Introduction

Even today, rain forecasts are still very computationally complex and relatively inaccurate. Therefore, it may make sense to make such predictions with the help of neural networks. These do not require as much computing power and can recognize a pattern in the often chaotic data even without complex physical models.

Data

Images must be generated from the binary radar data provided by the German Weather Service (DWD) in five-minute resolution. To convert the radar data into an image, each radar data point must be transformed into a pixel color value. In order to achieve a suitable transformation, the radar data of June 2016 were inspected exemplarily (figure 1).

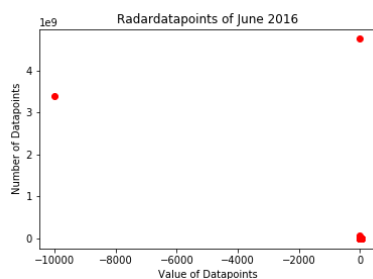


Figure 1: Frequency distribution of rain data

All occurring values and their frequency are plotted. Two outliers stand out in figure 1, which occur much more frequently than the other values. The value -9999 indicates that no data is available and the second outlier is at 0, which stands for no rain. In Figure 2 the two outliers are filtered, because the relevant information is contained in the remaining data points. In addition, the 99% percentile illustrates the frequency distribution.

The 99% percentile contains 138 different values, each of which is assigned a color

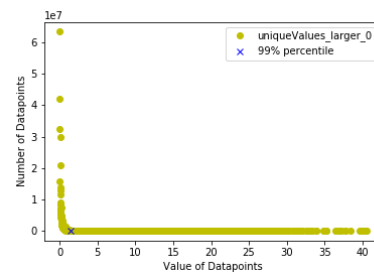


Figure 2: Frequency distribution of rain data with 99% percentile

value. The remaining data is transformed linearly to the remaining range of values. This results in rounding and maximum value errors. These errors are bearable, because it is more important to map all data points. Furthermore the resolution of the different rainfall intensities is still higher than the human perception. Reconstructing the radar data from the PNG data results in the plot shown in figure 3.

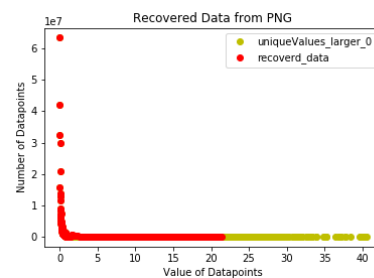


Figure 3: Comparison between output distribution and distribution from recovered data

Data Validation

To validate the quality of the data as well as the correct position of the constant pixel, the radar data were compared with the weather station in Konstanz. The radar data of one week from all over Germany were correlated with the weather station. The correlation coefficient reached its maximum around Konstanz. There are deviations from radar data to rainfall compensation, especially with small rainfall values, but these are caused

by inaccuracies of the weather station itself (figure 4). The correlation is significant, the position of Konstanz could be validated and the data quality is suitable for a rain forecast.

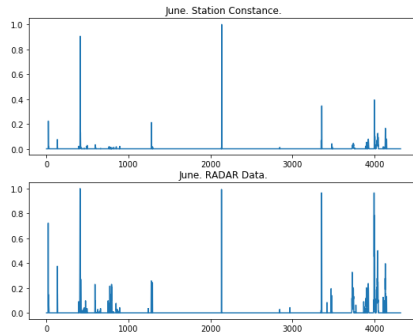


Figure 4: Comparison between Radar and Weatherstation in June 2016

DeepRain Application

The DeepRain app was developed with Flutter, which makes the app available Android and IOS. Firebase is used for the user administration, the forecast image transmission and the sending of rain warnings (push notifications). The serverside functions are written in python and the cloudfunctions, which are used for the push notifications, is written in JavaScript. The main code of the application is written in Dart. The App basically consists of three screens. One screen is the settings. In the settings you can change the region, configure the rain warning function and some other settings. On the Rain Map screen the current and future rain situation is displayed using the forecast PNGs. With a slider you can display the rain situation at different times. On the Prediction List screen the current and future rain situation is displayed quantitatively. The rain values are read from the respective pixel of the region in the forecast image.

Networks

We compare the performance of two neural network architectures. The first one is the widely known unet-architecture [RFB15] the second is a mixture of inception layer and convolution-LSTM which is derived from the paper "Inception-inspired LSTM for Next-frame Video Prediction" [Hos+19]. The output of our network is a 64x64 matrix of distributions. We compare a independent zero-inflated negative binomial distribution with a independent seven-class multinomial-distribution. The last layer (Ki-Schicht in figure 5) of both networktypes holds three vectors of dense-layers which will compute the parameters for one of the two distributions. Our baseline which we outperform assumes that the weather will not change in 30 minutes.

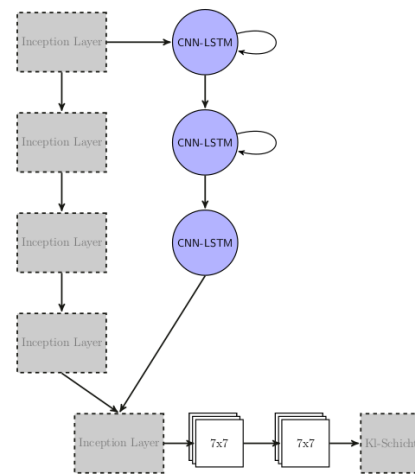


Figure 5: Our second architecture

For the binary rainprediction the receiver operating characteristic (more precisely the area under curve shown in table 1) concludes that the multinomial-distribution performs a little better than the negative binomial-distribution.

	LSTM		Unet	
	ZNIB	Multi	ZNIB	Multi
auc	64	68	66	69
NLL	0,82	0,36	0,94	0,37

Table 1: area under curve and negative loglikelihood loss

Table 1 also shows the best test losses which concludes the superiority of the multinomial-distribution. But when we review the rain intensity we see that no network architecture has a clear advantage over the other. Nevertheless as with the auc, we can determine slightly better performances. Although the overall accuracy is not very good we see in table 2 that the Unet-version performs the best. Note that the output of the zero-inflated negative binomial-distribution (ZINB) are discrete values between 0 and 255 (for prediction we use the mean of the distribution). We bin the values to seven classes so we can compare the distributions.

	LSTM		Unet	
Class	ZINB	Multi	ZINB	Multi
0	97 %	97 %	98 %	97 %
1	10 %	0 %	10 %	0 %
2	42 %	21 %	41 %	23 %
3	30 %	33 %	34 %	34 %
4	5 %	21 %	7 %	20 %
5	0 %	8 %	0 %	6 %
6	0 %	0 %	0 %	0 %

Table 2: Classification accuracy

Conclusion and Future Work

Both neural networks exceed the performance of the baseline. Nevertheless an improvement of the nets is still desirable. The pipeline from the server to the app runs stable and newly developed networks can be integrated with minimal effort.

References

- [RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *CoRR* abs/1505.04597 (2015). arXiv: [1505.04597](https://arxiv.org/abs/1505.04597). URL: <http://arxiv.org/abs/1505.04597>.
- [Hos+19] Matin Hosseini et al. *Inception-inspired LSTM for Next-frame Video Prediction*. 2019. arXiv: [1909.05622](https://arxiv.org/abs/1909.05622) [[cs](#),[CV](#)].