Improving rainfall rate estimations with Commercial Microwave Link Signals

in Sri Lanka using Deep Transfer Learning

Ludo Diender

Supervisors: Hidde Leijnse, Ruben Imhoff and Kirien Whan

February 2022

MSc thesis Hydrology and Quantitative Water Management Group
Wageningen University

Abstract

Don't make the abstract too long. See the list of tricks on Blackbord.

Contents

1 | Introduction

1.1 Context and motivation

Having ample and correct precipitation data is important for a plethora of applications, including flood warnings, agriculture, river safety and shipping routes (Chwala & Kunstmann, 2019). In urban areas, an even higher spatial and temporal resolution of rainfall is needed, due to the complex and quickly responding urban hydrological system (Overeem et al., 2011). A high rain gauge density can help in providing this resolution in urban areas (Yoon & Lee, 2017), but is not always availabe. Commercial Microwave Links (CML) do typically have a high density in populated areas and could therefore help in retrieving these high-resolution urban precipitation data. CML are back-haul links used by telecommunication companies to transfer information from one telecom station to the next using microwave signals. The links' signals get attenuated by rainfall by means of scattering and absorption. The received signal level is measured and stored by the telecommunication companies for monitoring purposes and can be used to retrieve path-averaged rainfall rates by calculating the attenuation. In the past 25 years, CML have been recognized as a valuable opportunistic method to measure rainfall (Leijnse et al., 2007; Ruf et al., 1996). Especially in data-scarce areas, where little precipitation is measured, CML have proven to be an excellent additional information source for precipitation data (Overeem et al., 2021; Doumounia et al., 2014; Diba et al., 2021). Recent research has been focussing on the use of CML data to measure rainfall in tropical regions, amongst others Sri Lanka (Overeem et al., 2021) and Brazil (Gaona, Overeem, Raupach, Leijnse, & Uijlenhoet, 2017a). Although operational use of CML signal as precipitation data source is mainly limited by the availability and accessibility of the signal data, the technique has been widely researched.

The first studies on the use of CML signals to retrieve rainfall rates were done by using a specific Power Law (PL) to relate the attenuation of the signal and the rainfall rate (Overeem et al., 2011; Leijnse et al., 2007).Several methods have been constructed based on this PL, including RAINLINK (Overeem et al., 2011). The RAINLINK methodology can be described with 4 steps:

1. wet-dry classification
2. baseline estimation
3. wet antenna attenuation estimation
4. rainfall rate retrieval

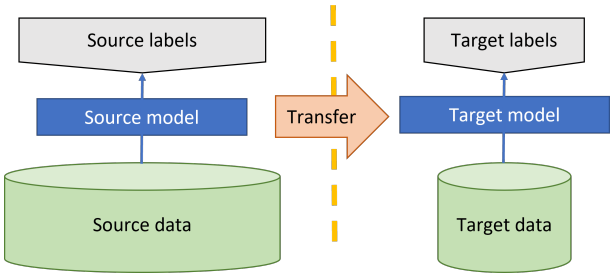

Figure 1.1: Concept of transfer learning. Adapted from (Sarkar, 2018)

This method has yielded good results in multiple studies (de Vos et al., 2019; Graf et al., 2020; Fencl et al., 2017). Recently, the community has been exploring a more data-driven approach in the form of neural networks of different architectures. Neural networks are a subpart of Machine Learning that is inspired by the neural structure of the brain. A neural network consists of different layers of nodes (neurons) that are connected to each other. If the network contains more than one layer, it is considered a deep network and Deep Learning is used as equivalent naming. A neural network is able to learn the relationship between the input and output, without intervention from a researcher. Studies have been performed in Sweden, Israel (V. Habi, 2019), Germany (Polz et al., 2020), South Korea and Ethiopia (Diba et al., 2021) on the use of such data-driven networks in relating CML signals to rainfall rates. Most of these use RAINLINK or a similar method as a benchmark. Previous studies have shown that data-driven models can be more accurate, less time-demanding and more robust in estimating rainfall rates compared to the PL method (Polz et al., 2020; Pudashine et al., 2020). Neural networks are not a novelty in predicting rainfall (French et al., 1992), but the application to CML data has recently gained popularity.

The few studies that have been performed on the combination of neural networks and CML, all focussed on a specific part of the RAINLINK methodology. Polz et al. (2020) and H. V. Habi and Messer (n.d.) put emphasis on the wet-dry classification. Pudashine et al. (2020) based his work around rainfall rate retrieval and Habi V. Habi (2019) constructed a combined model with both aspects present. All of these studies use a certain

amount of preprocessing and error filtering, aside from the deep learning, to deal with the other aspects of the original PL algorithm.

In focussing on a specific part of this method, the models can get more and more elaborate and complicated. Neural networks or deep learning can be a very versatile and powerful tool, but it comes at the expense of understandibility. The models can be tuned very specifically to the goal of the study, but it takes more and more time to understand the reasoning behind the final modelling output. An overarching, less complex and inclusive effort to model all steps of the RAINLINK algorithm at once has not been performed yet. This study aims to make use of the power of neural networks to deal with multiple steps in the RAINLINK algorithm at once. A deep learning effort to take all these steps into account has not yet been performed to the best of my knowledge.

1.2 Research questions

This study focusses on two main research questions. 1) How does a Neural Network perform on estimating rainfall rates from Dutch Commercial Microwave Link data by replacing all steps at once of the RAINLINK algorithm? 2) What model (hyper)parameters have the largest influence on this performance? By answering these questions, this study will give a first impression of the raw power of neural networks in combination with CML signals. It gives insight in what the most important factors are to take into account.

1.3 Thesis contents

This thesis starts with a description of the data that is used (Section 2). In Section 3 the methodology of this study is described. This includes an explanation on Neural Networks, data preprocessing, transfer learning and the set-up of this model study. Afterwards, in Section 4 the results are presented. These results and the approach used in this study are discussed in Section 4.3, which is followed by the conclusion in Section 5.

2 | Field site and data

Some text here about how the field site and data are structured? YES! DOI van dataset als hij op de 4Tu website staat!

2.1 Commercial Microwave Link data

2.1.1 Netherlands

Location of the links First paragraph text text text text text text text text text text text text text text text text text text text text text text text text text text text

Static vs dynamic data First paragraph text text text text text text text text text text text text text text text text text text text text text text text text text text text

Freq,temp res and timespan First paragraph text text text text text text text text text text text text text text text text text text text text text text text text text text text

2.2 Reference precipitation data

2.2.1 Netherlands

Obtained from (provider) First paragraph text text text text text text text text text text text text text text text text text text text text text text text text text text text

Temporal and spatial resolution First paragraph text text text text text text text text text text text text text text text text text text text text text text text text text text text

Quality and availability First paragraph text text text text text text text text text text text text text text text text text text text text text text text text text text t text

3 | Methods

This chapter describes the methodology of this study. The chapter starts with a general explanation on neural networks and their functioning, to get acquainted with terms and parameters in Section 3.1. Section 3.2 deals with the preprocessing of the input data to the model and subsequently model selection is described in Section **??**. Which model runs and how their results are evaluated will be treated in Section 3.4. The final part of this chapter deals with transfer learning and how this is applied in this study (Section **??**)

3.1 Neural network explanation

3.1.1 Goal and concepts of neural networks

Neural networks are a subset of models within the Machine Learning (ML) environment. All models within the ML space are data-driven, i.e. without pre-specified relationships between input and output. These data-driven models aim to figure out this relationship via self-learning techniques. The biggest advantages of such models are that 1) they are very versatile: they can adapt to various circumstances and can be used in a wide range of applications, and 2) they can learn relationships that we do not have a physical explanation for yet. Through multiple iterations, the models can improve by minimizing or maximizing a loss function. The loss function can be interpreted as a score for the model: the worse the score, the more the model needs to adjust itself (see Section 3.1.3).It improves by optimizing parameters of differential equations in the model. Similar to physically-based models, it is up to the modeler to determine when the model performs well enough. The self-learning ML models will stop learning after a certain criterion is met, or when a specified number of iterations has passed.

Neural networks are inspired by the structure of the human brain with neurons and synapses, hence the name neural network. The network consists of one or multiple layers of nodes that are connected with each other. When the network has more than one layer, it is called a deep neural network and it is referred to as Deep Learning. Every single connection between nodes (or between nodes and input/output) is represented by a weight and a bias. These weights and biases are the bread and butter of neural networks. They determine the underlying relationships between different parts of the input signal, the connections within the model and the relation to the final output. Training a neural network is all about adjusting the weights and biases in such a way that the desired output is created. At every node, a weighted sum is calculated based on these weights and biases and it is passed to an activation function. This activation function yields one value, which will be passed to the following nodes in the model. Because of the very general structure of neural networks, they form a very versatile set of models. Research efforts in neural networks span from image classification and speech recognition to time series prediction and weather forecasts.

3.1.2 Neural network architectures

The versatility of neural networks allows for a wide range of different model architectures to choose from. Every architecture has its own strengths and weaknesses. For image recognition for example, the most common type of neural network is a convolutional neural network (CNN). In time series prediction, the most common architecture is the recurrent neural network (RNN), which is therefore most suited for precipitation estimation using Commercial Microwave Links.

RNNs are looping-based architectures, which make them ideal for dealing with sequences. The core unit of RNNs, the Recurrent Unit (the lightgreen blocks in Figure 3.1), is used multiple times. The output of the unit is added to the input of the next step in the same recurrent unit. This creates an architecture that is well suited for time series with sequential data patterns. How many layers of these recurrent units are used in a model depends on the modeler and can be tuned (see Section 3.3)

One of the limitations of default RNNs is the lack of memory in the model. Information from the start of a sequence is quickly lost throughout the model. A Long Short Term Memory (LSTM) model circumvents this problem by having an extra stream of information. This stream of information carries aspects of the previous time steps and can be viewed as a 'memory highway'. Every layer of the network has a separate memory stream which carries information from the start of the sequence. At every LSTM cell, information is taken from the memory state as input to the cell, and an adapted memory state is given as output. Most cells in an LSTM model have three input streams: the memory state, the new input state and the hidden output state from the previ-

ous time step. Within each cell, the three input streams are used to produce three output states as well: the updated memory state, a hidden output for the next cell in the sequence and a hidden output for the next layer in the model. The conversion from these three inputs to three outputs is where the weights and biases come into play in an LSTM model. As an LSTM cell is a recurrent cell as well, there is one cell with weights and biases per layer of the network. Information is recurrently added to this cell. A schematic representation of the model used in this study can be found in figure 3.1.

After the information has passed through all LSTM layers of the model, the final hidden outputs are fed into the fully connected or dense layer (see figure 3.1). This layer takes the intermediate output from the LSTM cells and connects them to the desired output of the model. The dense layers outputs a sequence of values, the same length as the input sequence. In this study, a many-to-one architecture is used. This means that a sequence of signal levels is used to predict one rainfall rate at a specific timestep. Referring to figure 3.1, only the rightmost output (O_n) is considered the output of this model.

For more information on the internal functioning of LSTM cells, a more in depth explanation of their power and their evolution to their current shape and form, the reader is refered to Staudemeyer and Morris (2019). The LSTM approach in this study is similar, although not identical, to V. Habi (2019).

3.1.3 Learning process of neural networks

The self-learning ability of neural networks is mainly relying on the loss function. The model calculates the error between its output and the target output and tries to adjust accordingly. The selected loss function determines amongst others the behavior of the model. In this study, a standard Mean Square Error (MSE) loss function is used as a basis (see Eq. 3.1).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2 \tag{3.1}$$

where y_i is the predicted 15-min interval precipitation and $\hat{y}$ is the observed 15-min interval precipitation. MSE as a loss function has been used before in similar studies (Pudashine et al., 2020; Diba et al., 2021). These studies, however, had a more elaborate preprocessing of the data to correct for the inherently skewed nature of precipitation distributions. V. Habi (2019) showed that the loss function can also be altered to correct for this skewedness by using a Rain Distribution Factor (RDF). In this research, the application of an RDF serves two purposes:

- Penalize negative predictions more to force the model into predictions bigger than or equal to zero. A random initialization can cause negative output which is physically impossible.
- Penalize the model more for higher target variables. This steers the model towards a better prediction of the actual precipitation events.

The RDF is defined in Eq 3.2,

$$RDF = \begin{cases} 3, & y_i < 0 \\ 1 - y_s e^{y_r \cdot \hat{y}}, & \text{otherwise.} \end{cases} \tag{3.2}$$

where y_i is the predicted rainfall rate, $\hat{y}$ is the observed rainfall rate and y_s and y_r are hyperparameters, set to 0.95 and -5 respectively in accordance with V. Habi (2019).

The combination of Eq 3.1 and Eq 3.2 leads to the final loss function used in this thesis (Eq 3.3)

$$Loss = \frac{1}{n}\sum_{i=1}^{n} RDF\ (y_i - \hat{y})^2 \tag{3.3}$$

To update and improve the model prediction, an optimizer (or propagation algorithm) is selected. Such an optimizer is a mathematical set of equations that determines how the model learns. In this study, the Adaptive Moment (ADAM) optimizer will be used. ADAM is one of the more recently introduced optimizers and has been shown to outperform other optimizations methods. (Kingma & Ba, 2014) In simple terms: the method is looking for an optimum (minimum or maximum) per weight. Based on this gradient, the weight is adjusted up or down to minimize the loss function. As calculating the full gradient field for all weights present in the model is a tedious task, ADAM is stochastic method, which means it takes a randomly selected subset of the data and uses that to calculate the gradients. This might lead to more steps, but the steps are less computationally intensive and typically lead to an improved performance. The ADAM optimizer is back-propagated. This implies that the final layer of the network is updated first and the learning process works itself back to the start of the model. In this study, the fully connected layer at the end of the LSTM is updated first before the model improves and adjusts the LSTM cells itself. A back-propagated

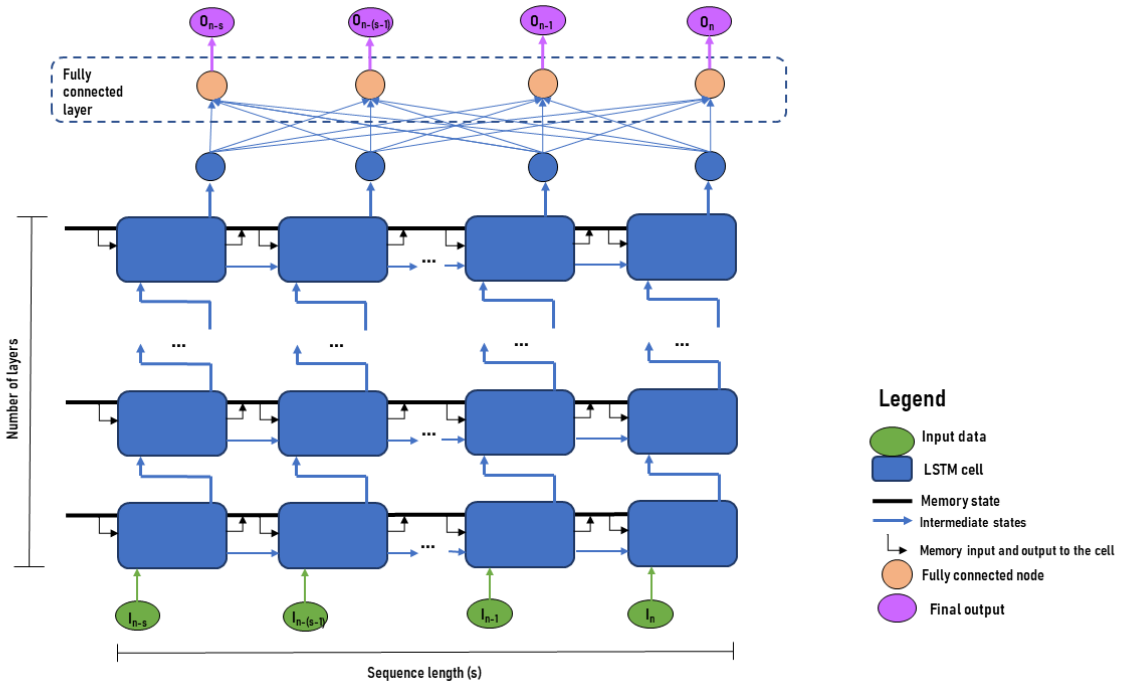

Figure 3.1: *Schematic representation of a two-layer unfolded RNN. HS represents the hidden state,X is the input, O is the output and U, V and W are weight and bias vectors associated with the model*

model is chosen in this study since it is the most common type of neural network, easy to implement, fast, flexible and generally good (Staudemeyer & Morris, 2019).

3.1.4 Available hyperparameters

The versatility of neural networks is generally perceived as one of the biggest advantages of these type of models. However, it does leave the modeller with a lot of options to choose from in creating a neural network for their specific task. The parameters that refer to the architecture and set-up of the neural network are called hyperparameters. Table 3.1 highlights the available hyperparameters in this model. The hyperparameters will, for the purpose of this thesis, be divided in three groups: Structure, Data and Effiency. Structure hyperparameters relate to the structure of the model: how is it designed and how often will it run. Data parameters are related to the input data into the model; the type of data and the goal with the data determine these hyperparameters. The Efficiency group entails hyperparameters that relate to the time it takes the model to run. In this group, there's always a balance between performance and computational intensity. It should be noted that in the end, all hyperparameters influence the computational speed of the model in some way or another. The proposed grouping is not strict but rather serves the purpose of dealing with hyperparameters with a specific function in a structured way.

The process of calibrating hyperparameters in order to find an optimal combination (if it exists) is refered to as tuning. Not all hyperparameters were automatically tuned. Only the Structure hyperparameters were tuned, as the other two hyperparameter groups need to be determined based on the input data and the available computational resources. More details on the automatically tuned Structure hyperparameters can be found in Section **??**. The Data and Efficiency group of hyperparameters are set based on convenience and literature study and will be elaborated on in Section **??**.

3.2 Data preprocessing

3.2.1 Connecting CML and reference data

To train the model, a feature set (received signal levels) and a target set (precipitation observations) is needed. For every separate CML link and every time step, the corresponding amount of precipitation has been determined. To obtain this value, the middle point of each link has been determined and connected to the corresponding radar cell. This midpoint method is commonly

Table 3.1: Available hyperparameters in this study.

Group	Hyperparameter	Description
Structure	Hidden size	Size of the intermediate hidden output. Higher hidden size allows for more complex relations
	Learning rate	A multiplication factor that influences the speed at which the weights in the model are updated with each SGD step. A high learning rate makes the model learn faster, but also makes it more prone to overfitting.
	Number of epochs	The number of iterations the model performs to reach the final prediction. More epochs improve the performance of the model but take sginificantly longer to run.
	Number of layers	Total number of layers in the model. A higher number of layers creates a model that can better identify complex dependencies in the data.
Data	Sequence length	Length of datasample used to predict one target value.
	Loss function	Metric that is used to judge the model performance on. Can be tailored towards the goal of the model.
Efficiency	Batch size	The size of each batch of data that is fed to the model. The larger the batch size, the faster the model will be able to learn, but the less possiblities it has to update its weights and biases.
	Optimizer	The algorithm that is used to update the weights and biases of the model internally.
	Scheduler	The algorithm that is used to run multiple combinations of hyperparameters simultaneously and decides when early stopping of trials is necessary

used in CML research (refs). More elaborate methods have been used by Leijnse et al. (2007) for example, where a weighted sum of all cells that the link passes through was created. The impact of this decision will be discussed further on in this thesis.

3.2.2 Error filtering

A large part of the RAINLINK algorithm consists of error filtering. Because the algorithm cannot distinguish a signal distortion by rain or by a bird, this filtering is necessary. The spirit of Deep Learning models is that these errors don't need to be filtered out manually. Similar to Habi et al., just the 'no data' values have been removed from the dataset. The model is expected to handle other noise in the data by learning the patterns and adjust accordingly.

3.2.3 Data splitting and scaling

A Deep Learning model with a varying set of hyperparameters needs three independent datasets to run: a training set, a validation set and a test set.The training dataset is used to improve the models weights and parameters and is fed through the SGD algorithm, in which the model is updated in a backwards propagating fashion. The validation set is subsequently used to independently judge the performance of the model. This set needs to be as independent as possible, to prevent overfitting of the model. As different models with different combinations of hyperparameters are tested, a third independent test set is needed to check how well the best model out of these different combinations performs.

In creating the three separate datasets, multiple approaches can be taken. The core of the split is based on the IID principle (Independent and Identically Distributed):

- There is little to no information leaked from one set to the other (truly independent datasets)
- There is enough data in each of the sets to cover a wide range of events and situations (comparable datasets and thus assumed identically distributed)

The data for the Netherlands has been split in a 40%/40%/20% for training, validating and testing respectively. Previous studies use varying ratios to split the data, with the one common aspect being a smaller validation set compared to the training and testing (Polz et al., 2020; Diba et al., 2021; Pudashine et al., 2020). This particular split has been chosen as a rough aver-

age of these varying ratios. The year 2011 is used for training, 2012 for testing and 2013 (up until July 21st) for validating. By taking full years for training and testing, all seasons are incorporated and the datasets are assumed to be closer to being identically distributed. Considerations on and implications of this split will be discussed later on in this thesis in section 4.3.1

Having large outliers and ranges in the input data can cause the Adam algorithm to have exploding gradients. Therefore, a scaling has been applied to both the features and targets. The targets (precipitation observations) have been scaled using a inverse hyperbolic sine transformation, which is a variant on the more common log transformation (Kilmartin & Peterson, 1972). To deal with zero values, which are ominous in precipitation data, and to handle extreme values better, Burbidge Burbidge, Magee, and Robb (1988) introduced the inverse hyperbolic sine transformation. This transformation is defined as in Eq 3.4.

$$y_t = \log(y_i + \sqrt{y_i^2 + 1}) \tag{3.4}$$

where y_t is the transformed variable and y_i is the raw variable. Afterwards, the logsine-transformed precipitation data were scaled with the minimum and maximum per dataset, to create a value between 0 and 1. As three separate datasets were used, this implies three slightly different scalings. As the goal is to minimize information leakage from one dataset to the next, this separated scaling is preferred, in order to not further violate the aforementioned IID assumption.

The features (RSL) have been scaled differently. As the base level of each CML is dependent on the frequency and the distance of the link, the median of each link has been subtracted from the signal. Subsequently, the data has been divided by the standard deviation per dataset to obtain values that are closer to eachother. By subtracting the median of each link from the signal, a very crude form of baseline estimation has been applied (step 3 in the RAINLINK methodology). As data needs to be rescaled anyways in order to avoid exploding or vanishing gradients, this specific scaling has been included in the methodology of this study.

3.3 Model selection

This section deals with selecting the right model structure for the study at hand. It is worth noting that there might be multiple combinations of hyperparameters that lead to the same results (a variant on the concept of equifinality). In section 4.3.2 this concept will be elaborated on and the impact on the followed procedure will be discussed.

Table 3.2: Ranges for the automatically tuned hyperparameters using RayTune.

Hyperparameter	*Range*
Hidden size	$[8, 32, 64, 128, 256]$
Learning rate	$1e^{-6} - 1e^{-3}$
Number of epochs	$(10 - 50)$
Number of layers	$[2, 4, 8, 16]$

3.3.1 Automated hyperparameter tuning

A full grid search on all hyperparameters available to find the best combination is computationally infeasible. Therefore, this study uses 25 random combinations of hyperparameters, all within a limited range of values. This number of combinations is chosen to have a trade-off between computational intensity and having ample combinations to ensure a decent search. Selecting the combinations is done using the RayTune package, which is a tuning-specific Python package that interacts with PyTorch (used in this study) and other Deep Learning packages. This study uses the Async Successive Halving Algorithm Scheduler (ASHA) to effectively search the hyperparameter space. The ASHA Scheduler is able to effectively implement parallelism (to decrease runtime) and a hard early stopping mechanism to terminate runs if they are not amongst the best few models after a while (Li et al., 2018). The ASHA Scheduler is the most common scheduler available in RayTune and performs well in most cases when no specific instructions regarding hyperparameter tuning are needed. The best combination of hyperparameters is used to calculate the model performance based on the final independent testset. The ranges for the calibrated hyperparameters can be found in Table 3.2

3.3.2 Non-automated hyperparameter tuning

Hyperparameters that are related to the data rather than the model itself are not tuned automatically. Hyperparameters in the Data group are tuned, but in a slightly less rigid way as it is easier to deduct which values should

perform well on the input data. The sequence length is partly determined by how long rainfall events typically last in the Netherlands and their autocorrelation. As a starting value, a sequence length of 8 (15-min) timesteps is used. This means that the past two hours are used to predict the rainfall rate at the current time step. A sequence length of one hour (4 timesteps) and three hours (12 timesteps) will be evaluated as well to inspect the impact of a different sequence length. The loss function has a direct relation to the distribution of the input data. The chosen loss function (3.3 should be able to deal with the skewedness of the data. To check if this is the case and whether the extra effort of including a RDF is worth it, a normal MSE loss function (3.1 will be tested as well.

From the Efficiency group, the batch size is the most important hyperparameter to evaluate.Kandel and Castelli (2020) described the effect of the batch size on the final performance of the model. The batch size and the learning rate are intimately linked: a smaller batch size requires a lower learning rate and vice versa. A batch size of 32 is used as standard, but the options of batch sizes of 64 and 128 are explored as well. A batch size of 32 has been shown to be a good default value (Bengio, 2012). Using a power of 2 as batch size is common to fully use GPU processing power.

3.4 Experimental setup and evaluation of results

For this study, different model runs are performed. In the first run, all automatically tuned hyperparameters mentioned in Section **??** are included. The 25 different randomly chosen model runs, as picked by the random selector in RayTune, are depicted in Table (INCLUDE TABLE WHEN ANUNNA IS READY). The model run with the best score on the independent test set will be used to test different non-automatically tuned hyperparameter settings. This results in three more combinations of model runs, where the batch size, the loss function and the sequence length are evaluated respectively with their ranges as put in Section 3.3.2. For the final model run, only timesteps that show precipitation are taken into account. This functions as a very crude wet-dry classification, the first step in the RAINLINK methodology. By running the model only on rain samples, we investigate the impact of using such a wet-dry classification, as multiple previous studies included a model part on this step (V. Habi, 2019).

The model results will be evaluated using the coefficient of determination (R^2, see Eq 3.5), the Coefficient of Variation (CV, Eq 3.6) and the Root Mean Square Error (RMSE, Eq 3.7).

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{N}(y_i - \bar{\hat{y}_i})^2} \tag{3.5}$$

$$CV = \frac{\sigma}{\bar{y}} \tag{3.6}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.7}$$

where y_i is the predicted 15-min precipitation rate, $\hat{y}_i$ is the observed 15-min precipitation rate and σ is the standard deviation of the predicted 15-min precipitation rate. These statistical evaluation metrics will be able to tell something about the fit of the model prediction to the observed values.2- The selected metrics are based on similar studies that use these metrics to evaluate their model performance (Overeem et al., 2011; Pudashine et al., 2020; Diba et al., 2021)

4 | Results

4.1 The Netherlands

Ideas: table with different hyperparameter sets and the best one out of all these. Line graph with 25 different lines: all the runs that are performed at once and to see the difference in their learning behaviour/where they end up. ONly the line that actually matters black, others can be greyed out.

Performance on training, testing and validation set for the best one

Scatterplot: predicted vs estimated precipitation for different loss functions, sequence lengths and batch sizes

Summarizing table with statistics on RAINLINK vs My Model

4.2 Transfer learning to Sri Lanka

Ideas: table with different hyperparameter sets and the best one out of all these.

Performance on training, testing and validation set for the best one

Scatterplot: predicted vs estimated precipitation for different loss functions

Summarizing table with statistics on RAINLINK vs My Model

4.3 Discussion

4.3.1 Data splitting

Including more information like neighbouring links The chosen years are not identical in their precipitation distributions. The year 2011 was somewhat drier than average, with a particularly dry spring. 2012 was slightly wetter than average, with above average precipitation in the first and last month of the year. Finally, 2013 had a dry winter with some snow and a very wet autumn, yielding a yearly average precipitation considerably lower than average. (SOURCE: KNMI?) This inconsistency in precipitation over the years violates the IID assumption that underpins the datasplit. It highlights the tradeoff that has been made in creating this split. Ideally, all datasets would have an equal precipitation distribution. However, wherever two datasets meet in time, there is a chance of information leaking from one dataset to the other. If the years would have been reshuffled to create a more identically distributed split, there would have been more information leaking from one set to the other and the independence of the datasets would have been at stake. Therefore, this split by full years was still preferred, to minimize information leakage. For the test set, half a year is taken due to limitations in data availability. As the validation set is only used to judge the final model, and not to update the model anymore, this dataset can be smaller as in (LITERATURE).

Include a part on discussing the impact of not having a valid IID assumption - multiple papers!

4.3.2 Equifinality and optimal hyperparameter combination

Trying out different types of NN and different hyperparameters: thousands to choose from

4.3.3 No signal picked up

More preprocessing of the data: stronger error filtering, only including the wet events (so wet-dry classficiation). Separate models with separate architectures for the different parts of the algorithm: a classifying CNN for the wet/dry, LSTM for baseline estimation, normal ANN for WAA and another LSTM for the final rainfall retrieval.

4.3.4 Outlook

Wat is de manier verder vanaf hier: wat zou wel kunnen werken?

5 | Conclusion

Does my model outperform RAINLINK in the NL? DL models have potential but there is a lot a lot to customize

Acknowledgements

Don't forget to thank the people who gave you data! T-Mobile Ralph Koppelaar & Ronald Kloeg Aart Overeem

References

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures [Book Section]. In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade: Second edition* (p. 437-478). Berlin, Heidelberg: Springer Berlin Heidelberg.

Burbidge, J. B., Magee, L., & Robb, A. L. (1988). Alternative transformations to handle extreme values of the dependent variable [Journal Article]. *Journal of the American Statistical Association*, *83*(401), 123-127. Retrieved from http://www.jstor.org/stable/2288929 doi: 10.2307/2288929

Chwala, C., & Kunstmann, H. (2019). Commercial microwave link networks for rainfall observation: Assessment of the current status and future challenges [Journal Article]. *WIREs Water*, *6*(2), e1337. doi: https://doi.org/10.1002/wat2.1337

de Vos, L. W., Overeem, A., Leijnse, H., & Uijlenhoet, R. (2019). Rainfall estimation accuracy of a nationwide instantaneously sampling commercial microwave link network: Error dependency on known characteristics [Journal Article]. *Journal of Atmospheric and Oceanic Technology*, *36*(7), 1267-1283. doi: 10.1175/JTECH-D-18-0197.1

Diba, F., Samad, M., Ghimire, J., & Choi, D. (2021). Wireless telecommunication links for rainfall monitoring: Deep learning approach and experimental results [Journal Article]. *IEEE*, *9*, 11. doi: 10.1109/ACCESS.2021.3076781

Doumounia, A., Gosset, M., Cazenave, F., Kacou, M., & Zougmore, F. (2014). Rainfall monitoring based on microwave links from cellular telecommunication networks: First results from a west african test bed [Journal Article]. *Geophysical Research Letters*, *41*(16), 6016-6022. doi: https://doi.org/10.1002/2014GL060724

Fencl, M., Dohnal, M., Rieckermann, J., & Bareš, V. (2017). Gauge-adjusted rainfall estimates from commercial microwave links [Journal Article]. *Hydrol. Earth Syst. Sci.*, *21*(1), 617-634. (HESS) doi: 10.5194/hess-21-617-2017

French, M. N., Krajewski, W. F., & Cuykendall, R. R. (1992). Rainfall forecasting in space and time using a neural network [Journal Article]. *Journal of Hydrology*, *137*(1), 1-31. doi: https://doi.org/10.1016/0022-1694(92)90046-X

Gaona, M., Overeem, A., Raupach, T., Leijnse, H., & Uijlenhoet, R. (2017a). Rainfall retrieval with commercial microwave links in sao paulo, brazil [Journal Article]. *Atmos. Meas. Tech.*, *11*(7), 11. doi: 10.5194/amt-11-4465-2018

Graf, M., Chwala, C., Polz, J., & Kunstmann, H. (2020). Rainfall estimation from a german-wide commercial microwave link network: optimized processing and validation for 1 year of data [Journal Article]. *Hydrol. Earth Syst. Sci.*, *24*(6), 2931-2950. (HESS) doi: 10.5194/hess-24-2931-2020

Habi, H. V., & Messer, H. (n.d.). Wet-dry classification using lstm and commercial microwave links [Conference Proceedings]. In *2018 ieee 10th sensor array and multichannel signal processing workshop (sam)* (p. 149-153). IEEE.

Habi, V. (2019). *Rain detection and estimation using recurrent neural network and commercial microwave link* (Thesis).

Hu, Q., Zhang, R., & Zhou, Y. (2016). Transfer learning for short-term wind speed prediction with deep neural networks [Journal Article]. *Renewable Energy*, *85*, 83-95. doi: https://doi.org/10.1016/j.renene.2015.06.034

Kandel, I., & Castelli, M. (2020). The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset [Journal Article]. *ICT Express*, *6*(4), 312-315. doi: https://doi.org/10.1016/j.icte.2020.04.010

Kilmartin, R. F., & Peterson, J. R. (1972). Rainfall-runoff regression with logarithmic transforms and zeros in the data [Journal Article]. *Water Resources Research*, *8*(4), 1096-1099. Retrieved from https://dx.doi.org/10.1029/wr008i004p01096 doi: 10.1029/wr008i004p01096

Kingma, D., & Ba, J. (2014). Adam: A method or stochastic optimization [Journal Article]. *arXiv eprint*. doi:

https://doi.org/10.48550/arXiv.1412.6980

Leijnse, H., Uijlenhoet, R., & Stricker, J. N. M. (2007). Hydrometeorological application of a microwave link: 2. precipitation [Journal Article]. *Water Resources Research*, *43*(4). doi: https://doi.org/10.1029/2006WR004989

Li, L., Jamieson, K., Rostamizadeh, A., Gonina, K., Hardt, M., Recht, B., & Talwalkar, A. (2018). *Massively parellel hyperparameter tuning (unpublished)* [Conference Paper].

Overeem, A., Leijnse, H., Leth, T., Bogerd, L., Priebe, J., Tricarico, D., ... Uijlenhoet, R. (2021). Tropical rainfall monitoring with commercial microwave links in sri lanka [Journal Article]. *Environmental Research Letters*, *16*. doi: 10.1088/1748-9326/ac0fa6

Overeem, A., Leijnse, H., & Uijlenhoet, R. (2011). Measuring urban rainfall using microwave links from commercial cellular communication networks [Journal Article]. *Water Resources Research*, *47*(12). doi: https://doi.org/10.1029/2010WR010350

Overeem, A., Leijnse, H., & Uijlenhoet, R. (2013). Country-wide rainfall maps from cellular communication networks [Journal Article]. *Proceedings of the National Academy of Sciences*, *110*(8), 2741-2745. doi: 10.1073/pnas.1217961110

Polz, J., Chwala, C., Graf, M., & Kunstmann, H. (2020). Rain event detection in commercial microwave link attenuation data using convolutional neural networks [Journal Article]. *Atmos. Meas. Tech.*, *13*, 18.

Pudashine, J., Guyot, A., Petitjean, F., Pauwels, V. R. N., Uijlenhoet, R., Seed, A., ... Walker, J. P. (2020). Deep learning for an improved prediction of rainfall retrievals from commercial microwave links [Journal Article]. *Water Resources Research*, *56*(7), e2019WR026255. doi: https://doi.org/10.1029/2019WR026255

Qureshi, A. S., Khan, A., Zameer, A., & Usman, A. (2017). Wind power prediction using deep neural network based meta regression and transfer learning [Journal Article]. *Applied Soft Computing*, *58*, 742-755. doi: https://doi.org/10.1016/j.asoc.2017.05.031

Ruf, C. S., Aydin, K., Mathur, S., & Bobak, J. P. (1996). 35-ghz dual-polarization propagation link for rain-rate estimation [Journal Article]. *Journal of Atmospheric and Oceanic Technology*, *13*(2), 419-425. doi: 10.1175/1520-0426(1996)013<0419:Gdpplf>2.0.Co;2

Sarkar, D. (2018, November 14, 2018). *A comprehensive hands-on guide to transfer learning with real-world applications in deep learning* [Web Page]. Retrieved from https://towardsdatascience.com/a-comprehensive-hand

Staudemeyer, R. C., & Morris, E. R. (2019). Understanding lstm–a tutorial into long short-term memory recurrent neural networks [Journal Article]. *arXiv preprint arXiv:1909.09586*.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., & Liu, C. (2018). A survey on deep transfer learning [Conference Proceedings]. In (p. 270-279). Springer International Publishing.

Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning [Journal Article]. *Journal of Big Data*, *3*(1), 9. doi: 10.1186/s40537-016-0043-6

Yoon, S.-S., & Lee, B. (2017). Effects of using high-density rain gauge networks and weather radar data on urban hydrological analyses [Journal Article]. *Water*, *9*(12), 931. Retrieved from https://www.mdpi.com/2073-4441/9/12/931

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., ... He, Q. (2021). A comprehensive survey on transfer learning [Journal Article]. *Proceedings of the IEEE*, *109*(1), 43-76. doi: 10.1109/JPROC.2020.3004555

A | Additional figures