Universität Augsburg
Fakultät für Angewandte Informatik
Institut für Geographie

Universität Augsburg
Institut für Geographie

Robustness of Precipitation Prediction using Machine Learning under Pseudo Global Warming

Bachelor Thesis

Alexander Mayer
M. Nr.:
B.Sc. Geography
Date of Submission: 12. March 2025

Supervision: Luca Glawion
First Examiner: Prof. Dr. Harald Kunstmann

Abstract

Data-driven weather forecasts are starting to reach and surpass the skill of physics-based ones. So far there is little research for using machine-learning based models under future climatic conditions.

This study evaluates how the performance of a ResNet-based precipitation forecast changes when input data is modified using a simple pseudo global warming approach. We trained three versions of the model using different loss functions on ERA5 reanalysis data. The models' performance was assessed using two different metrics for both the historical and the pseudo global warming data.

Our results show that all models produce comparable spatial characteristics, with a tendency to blur fine-scale precipitation features. They all significantly underpredict the total amount of precipitaton. The different loss functions differ in performance for different rain intensities, showing the need to deliberately select the appropriate loss function for the specific task.

The trained models keep the same spatial characteristics with pseudo global warming data, without an increase in blurring. Our model does not produce conclusive results for the forecast skill, suggesting the need for a pseudo global warming implementation with stronger physical constraints.

Keywords: Deep learning, climate change, loss function, precipitation forecasting, reanalysis, pseudo global warming, robustness

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CNN	convolutional neural network	1, 9, 11 f., 24, 34
DL	deep learning	1, 3, 9
ECMWF	European Centre for Medium-Range Weather Forecasts	1, 4
ERA5	ECMWF Reanalysis v5	1, 3–7, 24, 36
FSS	fractions skill score	3, 19 ff., 25 f., 30 ff.
GCM	general circulation model	15
LST	learnable sparse transform	35
ML	machine learning	1 ff., 5 f., 10, 12, 15, 21, 25, 28, 30, 36
MSE	mean squared error	15, 19, 23 f., 26, 34, 36
NWP	numerical weather prediction	1 f., 4
PGW	pseudo-global-warming	2 ff., 6, 15 f., 18, 21 f., 25 f., 28 f., 32 f., 36
RAPSD	radial averaged power spectral density	3, 20 f., 28, 30 ff.
TCWV	total column water vapor	32 f.
WMSE	weighted mean squared error	15, 23 f., 26, 28, 30 ff., 36

1 Introduction

Anthropogenic climate change with well over 1 °C of warming is affecting extreme events globally with large and varied negative impacts on the population and nature (IPCC, 2023:42). Unless rapid mitigation action is taken, global warming is predicted to intensify, in some scenarios in excess of 4 °C of warming (IPCC, 2023:65). The global water cycle is a key affected component, with dry events and seasons becoming dryer and wet ones becoming wetter, and the wettest-day precipitation on many areas of the globe increasing by more than 40% (IPCC, 2023:68-70). This shows the need for skilful weather and climate models in order to better understand these processes and their impacts.

So far numerical weather prediction (NWP) is the main approach to weather forecasting; ensembles of large numerical physics-based models are used to predict future weather, although this approach is limited by the immense computational power required (Ben Bouallègue et al., 2024:865).

1.1 State of the Art

Currently a breakthrough is happening in which several machine learning (ML)-systems surpass the performance of NWP. Pangu-Weather (Olivetti, Messori, 2024) achieves superior forecasts compared to the best NWP-system by the European Centre for Medium-Range Weather Forecasts (ECMWF) at more than 10 000 times the speed. Olivetti, Messori (2024) achieve this by using a neural network trained on ECMWF Reanalysis v5 (ERA5)-data. A different state-of-the-art ML forecasting model is GraphCast (Lam et al., 2023), which also uses ERA5-data to train a different type of neural network. AIFS is the data-driven forecasting system of the ECMWF (Lang et al., 2024) ready for operational use. Lang et al. (2024) use ERA5-data for training and have an encoder-decoder architecture, a general type of deep learning (DL).

Ben Bouallègue et al. (2024) describe how the release of ERA5 in 2020 has greatly improved the results of ML-based weather forecasts, as this is the best reconstruction of historical climate data so far. However, the quality of ERA5-data is not constant globally. Lavers et al. (2022) have found large deviations in the tropics, but satisfactory performance in extratropical regions.

Significant progress is not only made in general weather forecasting, but also explicitly in precipitation prediction, for which e.g. Badrinath et al. (2023) use convolutional neural networks (CNNs) to increase the forecast skill. While all of the previously mentioned ML models are predicting in a time frame of a few days, Gibson et al. (2021) have shown that by training on thousands of simulated seasons, ML can produce seasonal precipitation forecasts that outperform existing models.

Increasing the predictive period even further, Rackow et al. (2024) have tested the capability of state-of-the-art ML weather forecasting models – Pangu-Weather, GraphCast and AIFS – in different climates than the ones they were originally trained for; specifically in a pre-industrial, the current, and a 2.9 K warmer climate. While focussing on temperature, they also confirmed a high forecast skill for other variables. These results show great promise, keeping a good prediction skill with only some biases.

There exists also research on simpler, more accessible simulation methods. Brogli et al. (2023) describe pseudo-global-warming (PGW), a simulation strategy for regional climate models. Their contribution includes the Python package `PGW4ERA5`. One usage example is given by Heim et al. (2023), in which they simulated warming in the tropics.

1.2 Research Gap

Even though the current best models in AI-weather forecasting are substantially cheaper to run than those using NWP, they are still very large models requiring significant processing power. This is a limiting factor to trying different variations of the model design. These include e.g. the loss function, which pushes a model towards its objectives and can have several components, with a good example shown by Lang et al. (2024:5).

While Rackow et al. (2024) have analysed the robustness of these large models to climate change, a gap exists in this same type of analysis using lower-cost methods.

If simple PGW can be successfully used to drive the change in training data with measurable differences in model skill, this may enable the testing of different designs in low-cost models, which can then in turn be used in state-of-the-art models.

1.3 Objectives

Taking this context into account, we define the research goal of this thesis as follows: To investigate the robustness of machine learning models within climate change, specifically modelled using pseudo-global-warming (PGW). This goal includes two sub objectives:

First, the design of a machine learning (ML) short-term forecast model for precipitation. In particular we want to focus on investigating different loss functions that have a strong impact on ML model performance, especially in weather forecasts. We have chosen to develop our own model in order to make training on a single GPU feasible. The best models like AIFS and Pangu-Weather require high-performance computing clusters to run.

Secondly, the testing of the model's robustness when the input data is modified. Given the different definitions, we define robustness as the ability of the model to maintain its prediction skill for data increasingly far away from its training domain. For this, we use a simplified implementation of pseudo-global-warming (PGW). This is sufficient to answer our research question, as the goal is not to create the best possible prediction of a future climate, but to analyse the performance change of the models in a changed climate.

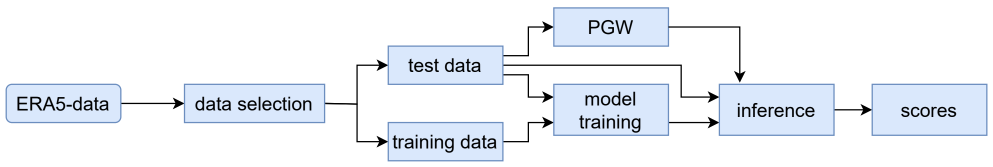

Figure 1: Simplified flowchart of the analysis.

An schematic overview of our study is shown in figure 1. The first step is the selection of ERA5-data, which we describe in section 2.1. In section 2.2 we process this data into samples and create two datasets, training and test data. We use this data to train a DL-model, which we describe in sections 2.3 and 2.4. In order to test the robustness as defined above, we modify the test dataset with a PGW-algorithm in section 2.5. Once the model is trained, we use it for inference – that is, prediction – on both test and PGW-data. To assess and quantify the prediction skill, we calculate the fractions skill score (FSS) and the radial averaged power spectral density (RAPSD), described in section 2.6.

The models weather forecast predictions are presented and evaluated in a qualitative and quantitative manner, using the FSS and RAPSD, in section 3. Finally, we discuss if the model is applicable under changed climatic conditions in section 4.

2 Data and Methods

In this chapter we present the used supervised machine learning model architecture, validation metrics and data processing steps are presented. The purpose of the model is to forecast precipitation fields with a lead time of six hours, from a time sequences of reanalysis images of previous time steps.

For this, as a first step, ERA5 data was downloaded. Secondly, the data was processed into samples to facilitate its use in the model, presented in section 2.1. Thirdly, we designed a neural network with the task of predicting precipitation and trained several versions of it, shown in sections 2.3 and 2.4. Additionally, we modified the selected data with a PGW algorithm, shown in section 2.5.

We performed all analysis in Python version 3.12.3, JupyterLab version 4.0.11.

2.1 ERA5 – Data Selection

ECMWF Reanalysis v5 (ERA5)-data forms the basis of this analysis. We will first give basic information about this data, before describing the choice of region, time and variables for downloading.

ERA5 is the current reanalysis product of the ECMWF, having replaced ERA-Interim in 2020 (Hersbach et al., 2020:2001). Reanalysis is the process of generating spatially and temporally continuous global climate variables by assimilating a variety of meteorological data into a NWP model (Gheysari et al., 2023:3088). This means that the past weather state is reconstructed in a way to fit historical measurements as good as possible. This approach has significant advantages to the conventional two primary sources of meteorological data for analysis: In-situ measurements are sparsely distributed and have no global coverage and often low spatial resolution, which means that interpolation is necessary between them (Gheysari et al., 2023:3088). Additionally, many weather stations do not provide long-term time series of data (Gheysari et al., 2023:3088). This effect can be seen in figure 2. Large areas of central Africa do not have any good long-term precipitation measurements. It should be noted that this does not directly impact ERA5, as no precipitation observations are used in the reanalysis, however the global distribution of meteorological observation stations is similar (Rivoire et al., 2021:1). Concerning the other common source of meteorological data, Gheysari et al. (2023:3088) point out that remote measurements in the form of satellite data also have issues as the data is often interrupted because of factors such as cloud coverage.

These disadvantages mean that for certain applications it can be advantageous to not use in-situ measurements or remote sensing, but instead reanalysis driven by these measurements. Examples are the data-driven weather models mentioned in the introduction like Pangu-Weather and AIFS, which both use ERA5. These benefit from the globally available, spatially and temporally continuous data, which is why we choose to use this data as well.

We acquired the data via the climate data store-API with version 0.7.0 of cdsapi. The product used is the hourly, single level reanalysis.

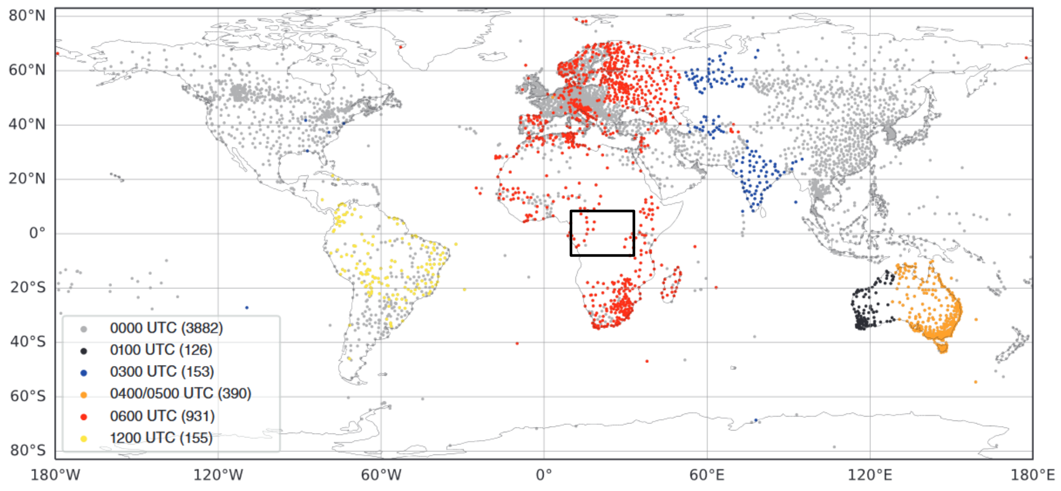

Figure 2: Distribution of gauge-based precipitation observations from the ECMWF archive with daily availability of more than 50% from 2001 to 2020. The total number is 5637. The area of interest of our study is marked. Adapted from Lavers et al. (2022:3154)

The selected area lies between 10°E, 33°E, 8°S and 8°N, as shown in figure 2. This is a rectangular area of around 2600 km by 1800 km, mainly encompassing the Congo Basin. The reasons for this selection are twofold: Firstly, each pixel in ERA5 when downloaded as NetCDF has an edge length of 0.25° which equates to 27.8 km at the equator (ECMWF, 2023). With increasing latitude, the distance between meridians decreases as the circumference of the earth decreases as well at that latitude. For pixels of fixed angular size, this leads to a smaller pixel area. This causes differing physical properties in the area of interest, e.g. a weather system of the same speed would move more pixels in a certain time frame at high latitudes compared to one near the equator. To avoid this difficulty, we chose an area around the equator, where differences in pixel size are negligible. The total area is thus 92×64 pixels.

Secondly, the choice of an area in the tropics mitigates one key problem of precipitation modelling: The data imbalance. The model benefits from an increased amount of precipitation events within the training data to learn the underlying patterns. In e.g. Central Europe, "no precipitation" is the far more common event. Learning reasonable predictions from such a minority class is a common issue for ML models

(Johnson, Khoshgoftaar, 2019:1). Frequent and abundant precipitation is a trait of tropical regions, effectively containing more information about it in the same amount of data.

ERA5 deviates significantly from real historical precipitation in the tropics, in particular tropical Africa (Rivoire et al., 2021:6). In particular Rivoire et al. (2021:10) assume ERA5 to underestimate extreme rainfalls in the tropics, though not as bad as its predecessor ERA-Interim. This is not a big issue, however, as in this study design the model is never applied outside of the training area and it does not affect the primary objective of analysing the robustness to change.

Due to resource constraints, six years of data were downloaded: 1980, 1981 and 2020, 2021 are used to select training data from, while the years 2000 and 2001 are used for test data and PGW. This temporal splitting routine is used to avoid information leakage from training to the test data, which otherwise might lead to hardly identifiable model overfitting. This is common practice in ML (Gudivada et al., 2017:5). By choosing test years inbetween the training years, we hope to minimise the impact of global warming on model performance. Half of the training data is from a slightly cooler, half from a slightly warmer climate compared to the test data.

We do not create a validation data set. This is normally used to adjust the hyperparameters in the training process (Freiesleben, Grote, 2023:108). However, as will be further explained in section 2.3, we skip this step in our model, as we instead use fixed values. This removes the necessity for a separate validation data set.

We selected five meteorological variables in total for training and testing our precipitation forecasting model, which are the surface variables as used in GraphCast (Lam et al., 2023:1). `tp` is the total precipitation and the quantity that will be predicted. It is the precipitation sum over the particular time period extracted with the unit m (ECMWF, 2025). In our case, this means $\mathrm{m\,h^{-1}}$, which we convert to $\mathrm{mm\,h^{-1}}$. The remaining four variables were selected , which we presume to have a large impact on precipitation. They provide the model with additional information, but are not part of the predictive model output, as we only train the model to forecast total precipitation. `u10` and `v10` are the two wind components at $10\,\mathrm{m}$ height. They represent the wind in easterly and northerly direction, respectively, and are given in $\mathrm{m\,s^{-1}}$ (ECMWF, 2025). `t2m` is the temperature two meters above the ground in K. Finally, `sp` is the surface pressure in units of Pa (ECMWF, 2025).

2.2 Generation of Processed Data Samples

Data driven methods depend on the provided input information in form of multi dimensional samples, which we generated randomly out of the ERA5 datasets. We define a sample as a spatiotemporal slice out of the entire dataset with a fixed size and length. The respective sets were loaded with xarray version 2024.11.0. We chose a spatial size of 10×10 pixels, i.e. $278\,\mathrm{km} \times 278\,\mathrm{km}$ to contain a sufficient number of features, while not being large enough for specific areas to be memorised. The most extreme case would be always training on the entire 92×64 pixel area. Missing the random component of sample locations, the model might learn an assignment of certain features to specific coordinates, rather than learning general patterns and relationships of variables. Furthermore, for a given number of samples, a larger size leads to higher demands on memory and computational capacity.

The second choice concerns the time frames selected of the hourly ERA5 data and consists of two parts. The first decision is how many hours of lead time t_1 to predict. The further into the future a prediction is, the worse its accuracy becomes (Ben Bouallègue et al., 2024:866). Really short lead times can be forecast by estimating the optical flow of moving rain fields, independent of the atmospheric dynamics (Ayzel et al., 2019:1399).

We do not want our model's capabilities to be limited to exploiting advection based information with the assumption of smoothly and persistent moving rain cells. Instead we want it to learn rainfall patterns, using all the different meteorological variables given. Therefore we choose a predictive lead time of six hours.

The second decision is how many preceding hours to include. The closer in time data is, the larger is its likely relevance for the prediction. However, a weather pattern might last weeks, with data thus having an impact for predictions weeks in the future. Here too a compromise has to be found due to resource limits. We chose a length of 48 hours – two days –, thus bringing the total selected time frame to 54 hours.

Each sample thus consists of five climate variables, 54 hourly steps and 10×10 pixels, with the the time frame being split into 48 hours of input data and 6 hours of target data.

We generated 10 000 samples for training and 1000 samples for testing from the already described years. The training sample size was the largest computationally feasible one. For each training sample we performed two checks: First, we reject all samples with insufficient precipitation, as those are dominant in the data set. By only selecting those samples with the maximum amount of information, the model can be trained more effectively (Magar, Barati Farimani, 2023:1). Some samples contain very little precipitation. We argue thus that they contain relatively little information about the underlying patterns. We selected them to satisfy the following equation 1 in which $n_{prediction}$ is the number of hours predicted and n_{width} the width in pixels.

$$\sum prec > 0.5\,\mathrm{mm\,h^{-1}} \cdot n_{prediction} \cdot n_{width} \tag{1}$$

The base value of $0.5\,\mathrm{mm\,h^{-1}}$ is chosen in order to further increase the amount of wet samples and the total amount of precipitation within the training dataset while still allowing a certain amount of dry and low precipitation events, which should not be completely removed.

The second check is for duplicate entries, as the random selection could select the exact same sample twice. Identifying and finding duplicates is essential (Gudivada et al., 2017:4). We rejected all samples with an identical spatial and temporal selection, replacing them until all were unique. Finally, all lists of samples were converted to numpy-arrays with numpy version 1.26.4. Next, we performed data cleaning. In the ERA5 dataset, zero-values for precipitation can be stored as very small negative numbers due to conversion errors, which we clipped to zero rainfall.

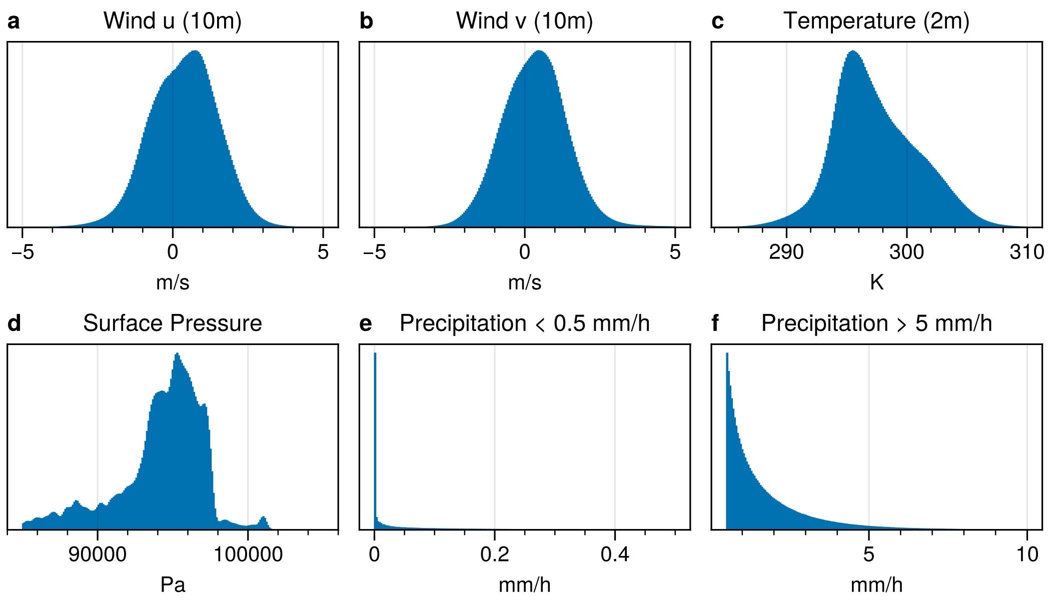

Figure 3: Schematic overview of the distribution of the individual variables of the samples within the training data set. Low precipitation intensities **e** and high precipitation intensities **f** are different sections of the same variable and do not share the same y-scale.

The distribution of the resulting training data set is shown in figure 3. The wind components (**a** and **b**) are slightly skewed to positive u and v direction. The surface pressure **d** has a very complex distribution. Of particular note is the precipitation distribution **e** and **f**. The majority class are pixels with the value of zero. This highlights the necessity of our deliberate sample selection routine by selecting a tropical domain of interest with an increased amount of rainfall and omitting samples with too little rainfall. If we did not do this, the distribution would be even more skewed. Far fewer pixels show a very low but positive precipitation of up to $0.1\,\mathrm{mm\,h^{-1}}$. Large values of e.g. $5\,\mathrm{mm\,h^{-1}}$ do exist in the dataset, but are a very small ratio of all pixels. The mean precipitation in this dataset is $0.253\,\mathrm{mm\,h^{-1}}$, the median is $0.009\,\mathrm{mm\,h^{-1}}$.

The final step of preprocessing is the normalisation of data. Different variables in the data can be of highly different scales, which leads to problems with gradient convergence (Wan, 2019:3). This problem will be explained in section 2.3. Min-max-scaling is a common method to scale the data to an interval of $[0,1]$ (Wan, 2019). Equation 2 scales a feature. The minimum value is assigned zero and the maximum value is assigned 1, with everything else stretched between them.

$$x_i' = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)} \tag{2}$$

We perform min-max-scaling on both the training and test data set, with one exception: We do not scale the total precipitation, as this variable is not only used as input for the model training, but also the target variable. This allows us to directly interpret the model output, without first having to undo the normalisation and rescale it to its absolute value. There is one essential modification to the scaling equation: We only use the minimum and maximum values of the training data set for all scaling. The possible consequence of values slightly smaller than zero or slightly larger than one is not an issue, as all data is still on a similar scale. This serves two purposes: Firstly a common scale is needed, as otherwise the test data would be distorted relative to the training data. Secondly, information only flows from the training data to the test data, not the other way around.

2.3 ResNet as Architecture of the Neural Network

To process the data we will use deep learning (DL). Artificial neural networks consist of many interconnected neurons which are able to learn patterns. A very simple example is shown in figure 4. Each connection between two neurons has a weight. It is through these weights that a specific output is calculated from an input. Deep learning (DL) describes a neural network with more than two layers, called deep neural networks (O'Shea, Nash, 2015:1).

A convolutional neural network (CNN) is a specific type of neural network primarily

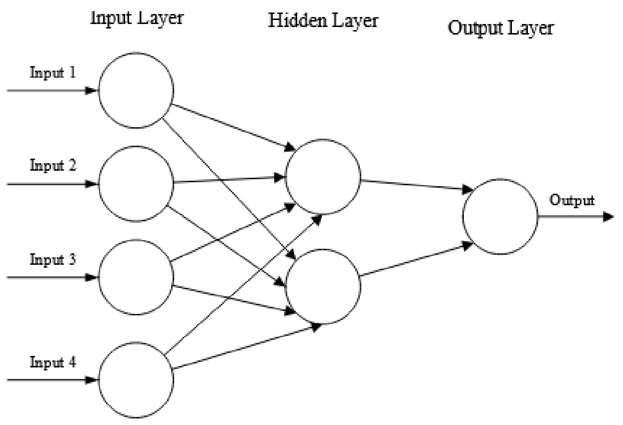

Figure 4: A simple feedforward neural network, in which neurons are connected in several layers, creating an output from several inputs. O'Shea, Nash (2015:2)

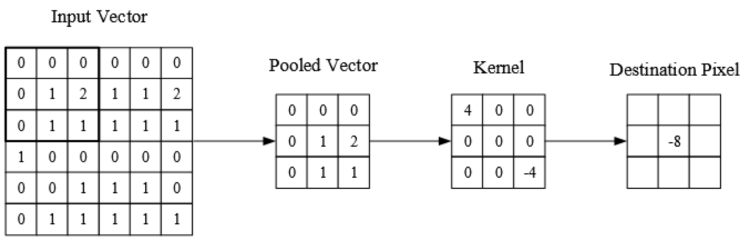

Figure 5: A simple example of 2-D-convolution. For a given kernel size – here 3×3 – a window of that size is moved over the input vector. For each window, here called pooled vector, the weighted sum is calculated with the kernel. This step is repeated for the entire input vector. O'Shea, Nash (2015:6)

used in image analysis (O'Shea, Nash, 2015:2-3). These are given their name through convolutional layers. The process of convolution is shown in figure 5 and is a way of incorporating information about neighbouring pixels into the output of that layer. During training, the model learns the weights of each kernel (O'Shea, Nash, 2015:5), which are essentially filters. In fully-connected layers, each neuron of one layer has a connection and thus weight with every neuron of the next layer, which is the main difference to convolutional layers, in which only a small region is connected, strongly reducing the model size (O'Shea, Nash, 2015:6). Apart from kernel size, there are two more parameters which will be relevant in our architecture: Stride defines the step size, with numbers greater than 1 creating a smaller image; Padding is added to the outside to allow the application of kernels on the outer pixels (O'Shea, Nash, 2015:7).

When designing a machine learning based precipitation field forecasting model, the problem at hand is that of predicting the next elements of a time series. Our model belongs to the area of supervised ML, in which the data for model training consists of input data matched with the intended output, the ground truth (O'Shea, Nash, 2015:2). More specifically, the first 48 hours of our data sets are used as input and the subsequent six hours are the ground truth, which the output is compared to.

Our architecture is based on a specific evolution of the CNN, the ResNet as published by He et al. (2015). This exists as a response to the increasing depth of networks – that is, the increasing number of hidden layers. Deep neural networks are generally better at their tasks, but after a certain number of layers the performance degrades again due to the difficulty of optimising the weights (He et al., 2015:1). He et al. (2015) address this problem by adding skip connections. These run parallel to other components of the network and are then recombined as the so-called residual. This eases the process of optimizing the weights, as the residual gives a good starting point.

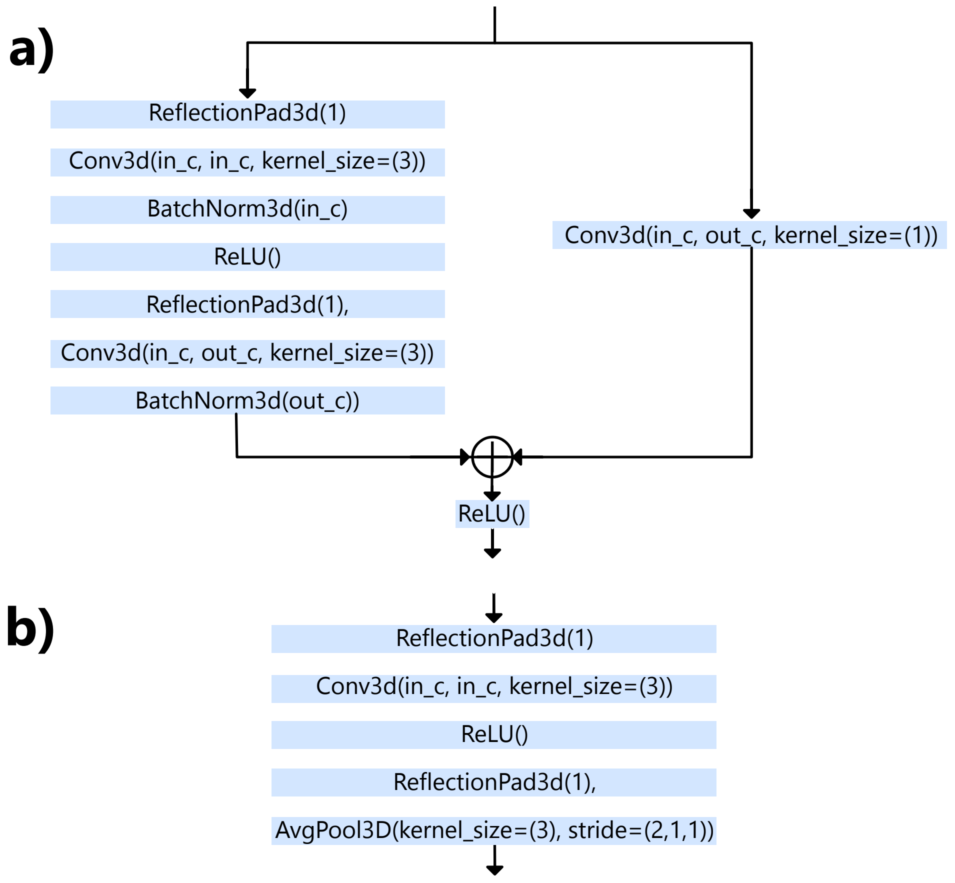
a)
ReflectionPad3d(1)
Conv3d(in_c, in_c, kernel_size=(3))
BatchNorm3d(in_c)
ReLU()
ReflectionPad3d(1),
Conv3d(in_c, out_c, kernel_size=(3))
BatchNorm3d(out_c))
Conv3d(in_c, out_c, kernel_size=(1))
ReLU()
b)
ReflectionPad3d(1)
Conv3d(in_c, in_c, kernel_size=(3))
ReLU()
ReflectionPad3d(1),
AvgPool3D(kernel_size=(3), stride=(2,1,1))

Figure 6: The two basic blocks out of which the neural net is built for a given number of input channels `in_c` and output channels `out_c`. **a** is the ResNet block, short ResBlock, and **b** is the convolutional block, short ConvBlock, which we use separately to the ResNet blcock. We use the function names of the pytorch implementation, but the mathematical operations are platform-independent.

Our analysis does not only evaluate and predict single, 2-D images, but also contains a time dimension. Therefore we use 3-D convolution for the temporal and the two spatial dimensions.

We performed our analysis using pytorch version 2.3.0 and cuda-runtime version 12.1.0. The main blocks of our implementation are shown in figure 6. We modified the core ResNet architecture in two ways, shown in the first diagram: Ioffe, Szegedy (2015:5) introduced the concept of batch normalisation in the model, which enables higher learning rates and deeper networks. Secondly, for skip connections we use a convolution of kernel size 1. This effectively means that each pixel of the residual is given a weight to be multiplied with, i.e. it allows scaling.

The second block is a simple convolutional block without a residual component. It contains an average pooling layer, which we use to downscale the time dimension.

For padding we have chosen to use reflection padding, which mirrors the values close to the edge. We did this to mitigate anomalies at the border which resulted from the previously used zero-padding.

The activation function is an essential element of CNNs by introducing nonlinearity to the layers (Li et al., 2020:4). This means that a neuron is only activated if the input is sufficiently large. The ResNet uses a very common activation function, ReLU – rectified linear unit (He et al., 2015:3). One advantage is its simplicity as it leaves positive values unchanged while replacing negative values with zero (Li et al., 2020:4).

We combine the two blocks shown in figure 6 into our entire architecture, presented in table 1. Of the four dimensions of our data, the first is the feature data. It starts at 5, which is the five different meteorological variables described in section 2.1. We increase this up to to 80, which allows the model to learn complex features, then finally down to 1, representing our output, the precipitation.

Regarding the temporal dimension, our input has a length of 48, which are the 48 hours that we use for the prediction. Using the ConvBlock and average pooling, we gradually downscale this dimension to arrive at a length of six hours, our forecast length.

Both input and prediction keep the spatial size constant at 10×10 pixels, equalling a size of 278 km $\times$ 278 km. We deliberately combine different convolution, average pooling and padding to maintain this size.

ML-models usually contain parameters that need to be set before training such as learning rate, so-called hyperparameters (Weerts et al., 2020:1). These are usually optimised using an extra validation data set, which we skip to limit the complexity of the model. Using default values for hyperparameters often does not result in

Table 1: Architecture of the ResNet. The layers are given in their correct order, with the first layer being at the top and the last at the bottom. The type refers to figure 6 in which they are described in more detail. The output shape are the dimension sizes of features, time, x and y. The number of parameters is the number of weights, which need to be optimised. The last layer is no complete full ConvBlock, instead just being a single layer. The total number of trainable parameters is 343 321.

Type	Output Shape	Param #
Input	(5, 48, 10, 10)	0
ResBlock	(10, 48, 10, 10)	2130
ResBlock	(20, 48, 10, 10)	8410
ResBlock	(40, 48, 10, 10)	33 420
ResBlock	(80, 48, 10, 10)	133 240
ConvBlock	(40, 24, 10, 10)	86 440
ResBlock	(20, 24, 10, 10)	65 800
ConvBlock	(10, 12, 10, 10)	5410
ResBlock	(10, 12, 10, 10)	5570
ConvBlock	(5, 6, 10, 10)	1355
ResBlock	(5, 6, 10, 10)	1410
Conv3D	(1, 6, 10, 10)	136

significantly worse performance (Weerts et al., 2020:9).

We have set the learning rate to 10^{-3} as used by Badrinath et al. (2023:295) and the batch size to 100, a common value. In the previous step, we generated normalised data sets. We transform those into pytorch tensors and generate a data loader using them, which makes them ready to be used for model training.

An important consideration in the design of a neural network is overfitting. This is the effect of the model having a reduced skill at learning generalised features, for example due to overly complex networks (O'Shea, Nash, 2015:3). If there is insufficient data relative to the complexity, the model will learn patterns of the sampling noise, which don't exist in other data (Srivastava et al., 2014:1929).

We have trained three different versions of our model, only differing by the choice of loss function, which will be described in greater detail in section 2.4.

All models were trained for 200 epochs. Each epoch consists of two steps: In the training loop the model is shown the entirety of the training data. The resulting predictions are compared to the ground truth, that is the final six hours of each sample. The difference between the observation and the prediction is calculated with the respective loss function. The weights are then updates in order to minimise the loss, which is done by an algorithm referred to as optimiser. For this we use Adam, which was published by Kingma, Ba (2015) and is commonly used, e.g. by Ayzel et al. (2020).

2.4 Use of Three Different Loss Functions

The purpose of loss functions in machine learning is to quantify the difference between the model output and the ground truth, with the mathematical goal of training being the minimisation of the loss (You et al., 2023:3). Careful selection of the appropriate loss function can significantly improve the training results (Tian et al., 2022:129). We refer to the output of the loss function as loss.

To contextualise the loss functions, we propose several effects of an ideal loss function: As seen earlier in figure 3, most pixels have a precipitation of $0\,\mathrm{mm\,h^{-1}}$, which should also be predicted as such. The total predicted precipitation sum should equal the target and large values should be correctly estimated. Extreme events should be reasonably predicted and a good prediction should preserve the gradients of the precipitation, as some precipitation events are very large with low gradients, while others have a high spatial variety with high gradients. Finally, the predicted rain cells should be at the right location, with minor deviations still acceptable.

However, machine learning models have difficulties to fulfil such requirements within their predictions. Common problems are the underestimation of extremes, leading to an underestimation of the total sum, such as in Polz et al. (2024) and Badrinath et al. (2023). Furthermore, many nowcast algorithms such as Harnist et al. (2024:3853-3854) provide overly smooth predictions with a loss of small-scale structures.

Within the "double-penalty error", uncertainties of the model and data lead to displaced rain cells, which is penalised twice (Necker et al., 2024:4457). One pixel should have a large value, but is predicted to be low, with the opposite being true on the pixel next to it. In this case, both pixels might contribute a lot to the total loss. This leads to an optimised model that is prone to underestimating extreme events, underestimating the total rainfall amount and generating rainfields that are too smooth.

To investigate how different loss functions contribute to these issues, we use and compare three different loss functions in total for our deep learning model. We will give the equations for the average loss for a prediction containing n pixels by comparing it to the observation.

As a baseline and first loss function we use the mean squared error (MSE) loss, a staple of ML in particular and statistics in general:

$$\mathrm{MSE} = \frac{1}{\mathrm{n}} \sum_{i=1}^{n} (\mathrm{pred}_i - \mathrm{obs}_i)^2 \tag{3}$$

To improve it, Sai et al. (2009) have proposed a modification in the form of weighted mean squared error (WMSE), in which the contribution of each individual error does not have the same proportion, but it weighted instead. We implement a simplified version of this, with every pixel with a value above a threshold being more heavily weighted.

$$\mathrm{WMSE} = \frac{1}{\mathrm{n}} \sum_{i=1}^{n} w_i (\mathrm{pred}_i - \mathrm{obs}_i)^2, w_i = \begin{cases} w & pred_i >= th \\ 1 & else \end{cases} \tag{4}$$

The intended benefit of this approach is to counteract the previously described problem of underestimation of large precipitation values, as this would produce a larger error compared to simple MSE. In preliminary tests we tried two combinations of weight factor and thresholds. First $w = 3, th = 0.5\,\mathrm{mm\,h^{-1}}$, then $w = 2, th = 1\,\mathrm{mm\,h^{-1}}$. We used the latter values for the full model training as these produced sharper predictions.

As a third loss function we use the Logcoshloss, which is one of the most important loss functions currently used in ML (Saleh, Saleh, 2024:1). It uses the hyperbolic cosine `cosh`. Ayzel et al. (2020) use this loss function in their neural-net-based nowcast RainNet, defining the function:

$$\mathrm{LogCoshLoss} = \frac{1}{\mathrm{n}} \sum_{i=1}^{n} \ln(\cosh(\mathrm{pred}_i - \mathrm{obs}_i)) \tag{5}$$

2.5 Implementation of Pseudo Global Warming

High resolution long term general circulation models (GCMs) have an immense computational cost (Schär et al., 2020:583-584). This means that it is not feasible to use them for a lot of research, where different techniques are used instead. Ban et al. (2021) limit their kilometer-scale climate models to a specific region, greatly limiting the computational power required. They use ERA-Interim data for the boundary conditions of their evaluation run. An alternative is pseudo-global-warming (PGW), where the boundary condition of the future climate is directly calculated from the control climate (CTRL), as given in the following equation (Brogli et al., 2023:908):

$$\mathrm{PGW} = \mathrm{CTRL} + \Delta \tag{6}$$

Specifically for this definition, CTRL and PGW are the boundary conditions of the past and future, with the difference being the so-called climate change delta Δ (Brogli et al., 2023:908). The precise implementation varies, but generally the Δ includes changes of temperature, humidity, wind and pressure and is computed from a separate climate projection (Brogli et al., 2023:908).

Unlike e.g. Ban et al. (2021), our goal is not the best possible simulation of a regional climate. Instead, our main objective is testing the robustness of our model to a shift in distribution, i.e. under climate change. For this reason, we can further simplify the PGW-approach without negatively impacting the objective.

While still using equation 6, our PGW differs in two ways: First, we do not compute Δ from a separate climate projection, instead directly increasing the temperature and deriving plausible values for the other variables using literature. Secondly, we do not only change the boundary conditions and then run a regional climate model with them, instead uniformly applying those changes across the entire historical data. We model the changes for different variables as follows:

For the AOI, no significant change in the mean wind speed is expected for the high emission RCP8.5 scenario in the far future (Jung, Schindler, 2019:5). This scenario corresponds to a warming of 5 K (Pielke Jr et al., 2022:1). The distribution of the wind speed will likely change in standard deviation – an increase of 10% with 5 K of warming –, skewness and kurtosis (Jung, Schindler, 2019:7-9), of which we only model the change in standard deviation to slightly simplify the needed statistics.

One key difference needs to be considered to implement these changes: While Jung, Schindler (2019) use the absolute wind speed without any information about the wind direction, our data contains positive as well as negative values, corresponding to different wind directions. This means we need to separately take all positive and negative wind values and increase their standard deviation without increasing the respective mean. We achieve this by subtracting the mean from each value, then scaling it, then adding the mean again.

To calculate the change of surface pressure, we combine two sources. Schmidt, Grise (2017:10575) calculate the surface level pressure change from 1980 to 2010 globally, which we estimate in the AOI to be around 0.1 hPa per decade. Temperatures on the African continent have increased by 0.28 K per decade since 1982 (NCEI, 2023).

We combine these two numbers to arrive at a pressure increase of 0.36 hPa per 1 K of warming.

Next we model the effect of warming on precipitation. Cook et al. (2025) analyse the effects of a future climate on the Congo Basin, our AOI. With a warming between 3 and 4.7 K, the seasonal mean of precipitation does not change significantly, but intense rainfall occurs more often and in unprecedented amounts compared to the current climate (Cook et al., 2025:381-383). To model this increase, we apply the well-known thermodynamical Clausius-Clapeyron relation, which describes an increase of the water-holding capacity of the atmosphere by about 7% for every K of warming. A strong relationship between the total column water vapor and the precipitation exists in the tropics (Trenberth, 2011:124). Additionally, high values of total column water vapor are a necessity for the unprecedented amounts of rainfall described by Cook et al. (2025:383).

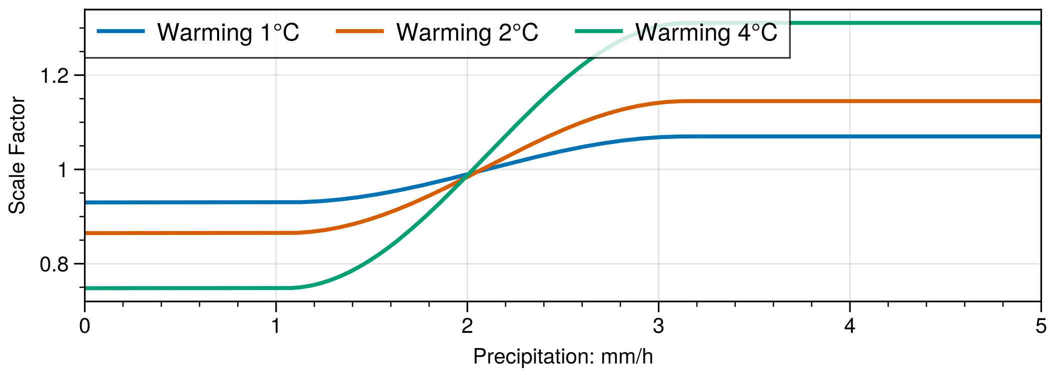

Figure 7: Scaling function to multiply the precipitation intensity with for a given amount of warming. The function is centred around 2.1 mm h^{-1}, which was the value chosen in order to keep the mean of the entire distribution constant.

We use a function that satisfies the following two conditions: The mean precipitation does not change significantly, but values above a certain threshold increase by 7% per K of warming. To counteract the effect on the mean, we decrease all low values similarly. However, having one linear scaling factor below a threshold and another one above with a discontinuity between them would lead to a gap in the histogram with values around the threshold missing. To avoid this, we combine the two straight lines with a segment of a sine function, thus gradually increasing the scaling. This equation is shown in figure 7 for several distinct values of warming.

$$p_{PGW} = p \cdot \begin{cases} 0.93^T & p < \frac{1}{2} p_0 \\ 1.07^T & p > \frac{3}{2} p_0 \\ \frac{1.07^T - 0.93^T}{2} \sin\left(\frac{\Pi}{p_0}(p - p_0)\right) + \frac{1.07^T + 0.93^T}{2} & else \end{cases} \tag{7}$$

The pixelwise distribution of precipitation intensity is extremely right-skewed, as seen in figure 8. This means that for the vast majority of pixels, equation 7 reduces the value. However, as this reduction is a percentage, the absolute reduction per pixel is small. There are far fewer pixels in which the precipitation strength is above the threshold. Conversely, the relative increase corresponds to a far higher absolute increase though. The precise value for p_0 such that the mean stays constant depends on the specific distribution of the sample. We have however found empirically for our test dataset that $p_0 = 2.1\,\mathrm{mm\,h^{-1}}$ is the value with minimal effect on the mean. This means that all values below $p = 1.05\,\mathrm{mm\,h^{-1}}$ are reduced to a factor of 0.93^{T} and all values above $p = 3.2\,\mathrm{mm\,h^{-1}}$ are increased to a factor of 1.07^{T}. For instance, a precipitation of $p = 5\,\mathrm{mm\,h^{-1}}$ under 3 K warming is calculated as $5\,\mathrm{mm\,h^{-1}} \cdot 1.07^3 = 6.1\,\mathrm{mm\,h^{-1}}$.

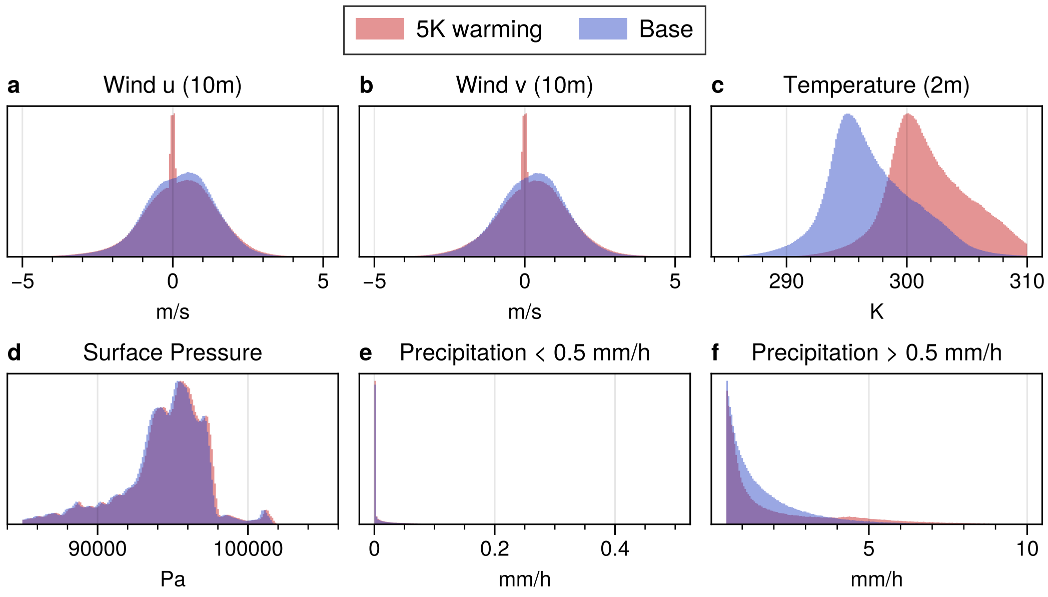

Figure 8: Histograms of the test data set and 5 K warming one. Low precipitation **e** and high precipitation **f** are different sections of the same variable and do not share the same y-scale. 5 K is shown here as the effects are most obvious, but a warming of lower strength, e.g. 3 K, shift the distribution in a similar but less intense way.

To show the effect of our PGW-algorithm, we have plotted the histogram of the most extreme warming used – 5 K – against the test dataset in figure 8. The most obvious change is the temperature, which is a direct shift. In addition small absolute wind values are moved closer to zero to counteract the increase of larger values, as the mean stays the same. Meanwhile it can be seen how large precipitation values are increased.

2.6 Validation Metrics

In section 2.4 we introduced the three different loss functions used for training our model. One might assume that these functions could be used to directly quantify the quality of rainfall predictions, but this is not the case.

Firstly, the resulting losses of different functions cannot be compared to each other, as is later shown in figure 11, as the specific loss depends on the function used. Loss functions might not be a good representation of what a good rainfall prediction is, for example due to the already described double-penalty error (Necker et al., 2024:4457).

For this reason we want to use two different validation metrics.

2.6.1 Fractions Skill Score

As a first validation metric we use the fractions skill score (FSS) as implemented in version 1.3.0 of the `scores`-package. Leeuwenburg et al. (2024) used a large number of functions in a variety of categories such as probability and categorical. For spatial analysis, they used the FSS as described by Roberts, Lean (2008). The goal of Roberts, Lean was to create a metric for forecast skill at different scales that is easy to understand by comparing the prediction to the observation.

The first step is the conversion of both the observation and prediction fields to a binary field (Roberts, Lean, 2008:80). First a threshold is selected, e.g. $1\,\mathrm{mm\,h^{-1}}$. Then the binary operation is performed in both the observation and the prediction: All pixels with a precipitation value greater or equal to the threshold are assigned 1, all others are assigned 0 (Roberts, Lean, 2008:80). Their next step is the generation of fractions as shown in figure 9. A window is defined over which the binary fields are compared with larger windows corresponding to a blurrier analysis (Roberts, Lean, 2008:81).

They perform this calculation over the entire prediction. Next they calculate the MSE over all individual fractions. Finally, the FSS is defined by comparing the resulting MSE to the MSE of a low-skill reference forecast (Roberts, Lean, 2008:82). A forecast without error is given a score of 1, while lower numbers correspond to worse forecasts:

$$\mathrm{FSS}_{(n)} = 1 - \frac{\mathrm{MSE}_{(n)}}{\mathrm{MSE}_{(n)ref}} \tag{8}$$

We use the FSS in our analysis to compare the prediction of a given model to the observation. We use three different thresholds: $0.1\,\mathrm{mm\,h^{-1}}$ to see how well

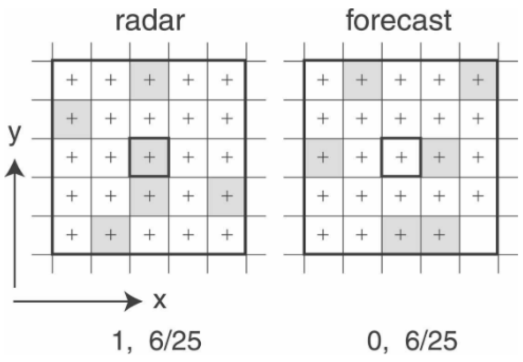

Figure 9: Schematic example of the fractions for the observation (left) and the forecast (right). All shaded pixels exceed the defined threshold. Although the forecast at the central pixel is wrong, over the entire 5×5 window the number of values exceeding the threshold is identical, which means this forecast is considered correct. Roberts, Lean (2008:81)

the predicted general locations of rainfall match, $1\,\mathrm{mm\,h^{-1}}$ for moderately higher intensities and $2\,\mathrm{mm\,h^{-1}}$ for even higher values. We use two different window sizes, 1×1 pixels and 4×4 pixels. These equal $27.8\,\mathrm{km} \times 27.8\,\mathrm{km}$ and $111.2\,\mathrm{km} \times 111.2\,\mathrm{km}$, respectively. We choose these values to compare the performance without any and with some spatial tolerance. The score is calculated for all samples in a dataset. As a final score, we take the mean of all individual FSSs.

2.6.2 Radially Averaged Power Spectrum Density

As a second metric we implement radial averaged power spectral density (RAPSD), which we use to analyse the spatial structure of our predictions. This is important as precipitation patterns exist in a large range of scales, from local thunderstorms to several 100 km large systems (Ruzanski, Chandrasekar, 2011:2296). Previously used in image processing, Ruzanski, Chandrasekar have applied this technique in their 2011 paper to the analysis of radar fields of precipitation. They summarise its effects as taking 2-D data in the frequency domain using a Fast-Fourier-Transform, converting it to polar coordinates and then removing the angular component through summary, thus getting a 1-D spectrum. This can be seen in figure 10. Ruzanski, Chandrasekar (2011:2301) combine the radial frequency f_r with the grid spacing s to calculate the wavelength:

$$\lambda = s/f_r \tag{9}$$

One application of RAPSD is the detection of smoothing, for which Harnist et al. (2024:3856) use it. At small wavelengths the RAPSD of their model outputs is significantly below the observations, which means that the forecast does not represent small-scale features well.

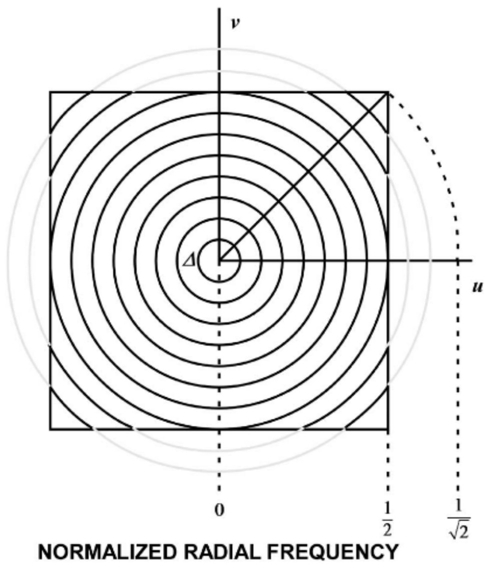

Figure 10: Illustration of the RAPSD. The 2-D frequency representation of an image is first converted to polar coordinates, with the origin at the centre. The circles represent radial frequencies which are used for a spectral analysis. Ruzanski, Chandrasekar (2011:2301)

We use the RAPSD-implementation of Pysteps, version 1.13.0. This is a Python library built for precipitation nowcasting (Pulkkinen et al., 2019). It uses a Fast Fourier Transform to calculate the power density for each radial frequency, but this is hard to interpret. We instead convert the frequency to a wavelength using equation 9. The grid spacing of the data used is 27.8 km (ECMWF, 2023).

As we calculate the RAPSD for entire datasets, our next step is calculating the mean power density per wavelength. The power density at larger wavelengths is usually several orders of magnitude larger than at small wavelengths, so we convert the values to a logarithmic scale to enable better comparisons.

Unlike FSS, this metric does not directly compare a prediction with the corresponding ground truth, but only calculates the power density for a specific sample.

3 Results

As described in section 1.3, our research goal is to analyse the robustness of ML models under climate change, specifically the effect of different loss functions. To achieve this we created and trained three ResNet-based precipitation forecast models with different loss functions. To test the robustness of the models under the impact of global warming, we modified our test data via simple PGW. Finally, we implemented different validation metrics to compare the prediction performances of different model-data combinations. Here we will describe those results.

3.1 Evaluation of Model Training Progress and Selection of Best Performing Model State

As described in section 2 we trained three deep learning models using different loss functions for 200 epochs each. The logged losses during training are shown in figure 11 for the training dataset and the test dataset, without PGW.

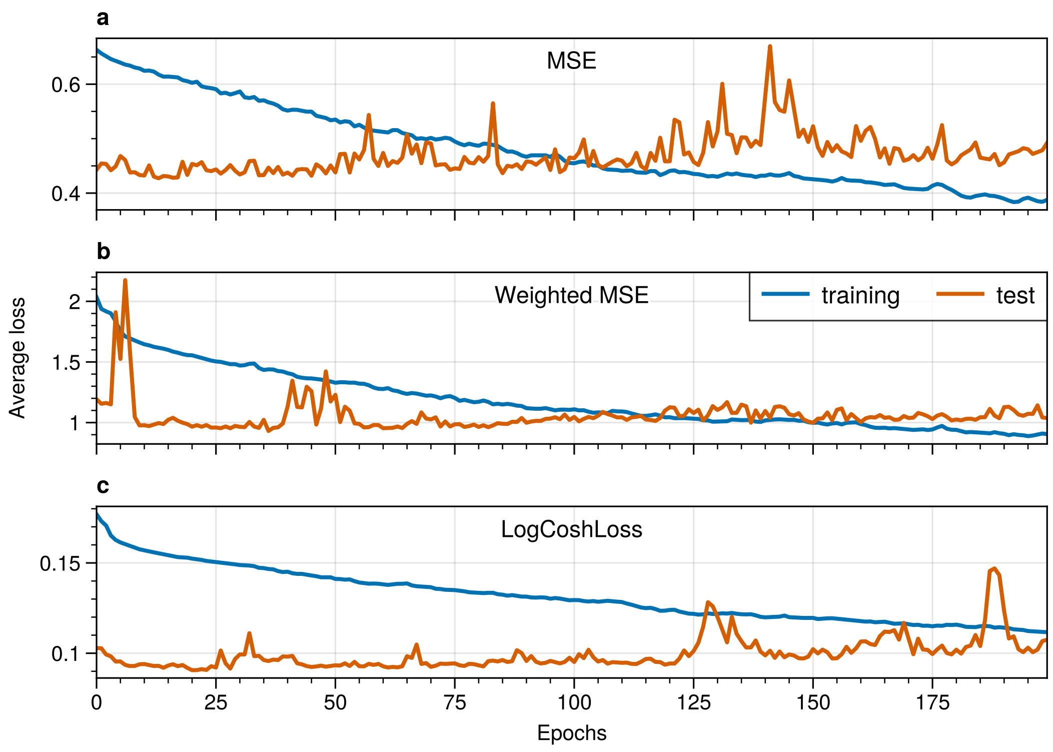

Figure 11: The training and test losses for the models trained with three different loss functions. The absolute loss values depend on the loss functions and the underlying training and test samples, and can therefore not be compared to each other. The best test loss of **a** was reached in epoch 13, for **b** in epoch 36 and for **c** in epoch 20. The training loss declines in all models, while the test loss slightly increases past the optimum, likely due to overfitting.

We track the losses for two reasons: First we want to know which model state is the best one for inference, i.e. prediction. We also want to know if overfitting occurs, which is typically indicated by a rise of test loss with increasing training.

The evolution of the loss during training is shown in figure 11. The training loss behaves very similarly for all three models with the loss declining nearly continuously, slightly faster at the start than towards the end of training. This is the expected result, as in each epoch the model sees the entire training data set and can optimise its weights for it, steadily learning a better representation of this data set.

Returning to figure 11, the test loss behaves far less predictable than the training loss. The goal is to train a well generalizing model that not only minimizes the training loss, but also performs well for the test dataset. The model is not trained on this test data, so it represents the ability of the model to generalise on previously unseen data. All three models reach their best test loss fairly early in training. For MSE it is epoch 13, for WMSE epoch 36 and for LogCoshLoss epoch 20, out of a total of 200 epochs. After these best states, the test loss rises in all models. This may be a sign of overfitting, which we will further test in section 3.3.

3.2 Qualitative Evaluation of Model Prediction

Before using the implemented metrics, we will first do a qualitative comparison of the three different models. For this, we will review the training progress of the model and compare different outputs.

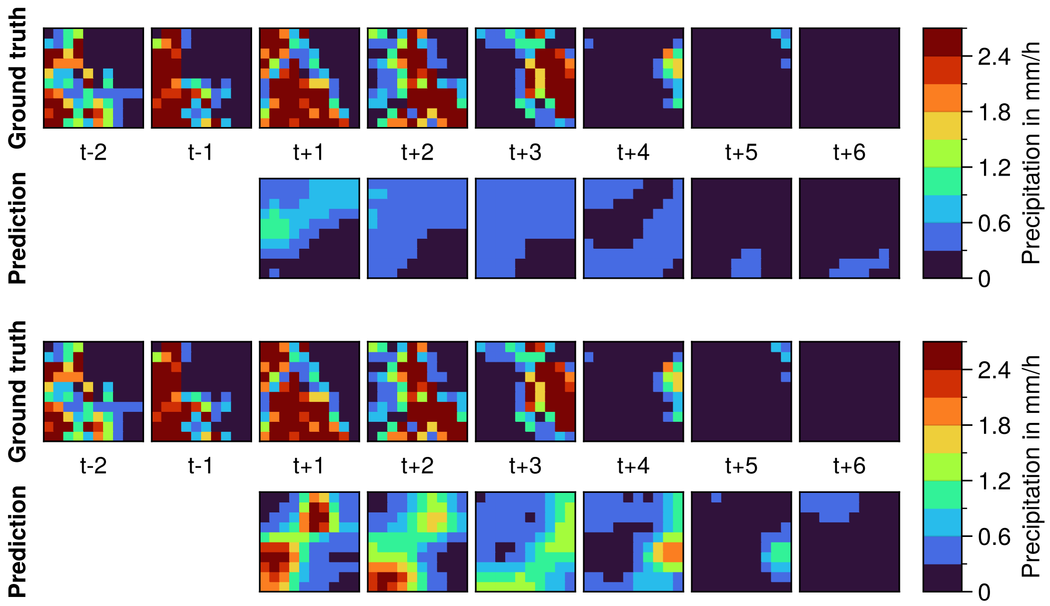

Figure 12: Model prediction of the same event compared to target in a 2.5° by 2.5° area. Shown is the training progress by comparing two outputs of the LogCoshLoss model with training data as input at epochs 10 (top) and 196 (bottom).

One example for this is shown in figure 12, in which the same sample out of the training data set is used for a prediction very early in training (epoch 10) and towards the end (epoch 196). The first prediction shows first learning progress: In hours with large precipitation patterns (such as the t+1 to t+3), the model outputs large areas of rainfall. Meanwhile in hours of low precipitation, t+5 and t+6, the model also predicts this correctly. It does, however, vastly underpredict the intensities and creates very blurry images.

The blurring of predictions is a known issue of deterministic precipitation nowcasts as the model tries to minimise the pixelwise error forecasts (Harnist et al., 2024:3840).

The second prediction in figure 12 after nearly 20 times as much training is much closer to the ground truth. It correctly shows much higher intensities and contains far better defined structures. At the same time it does still deviate from the truth, though to a lesser extent than the earlier model. We once again expect this behaviour, as CNNs first learn general features such as edges before being able to build complex structures in the image (Badrinath et al., 2023:301).

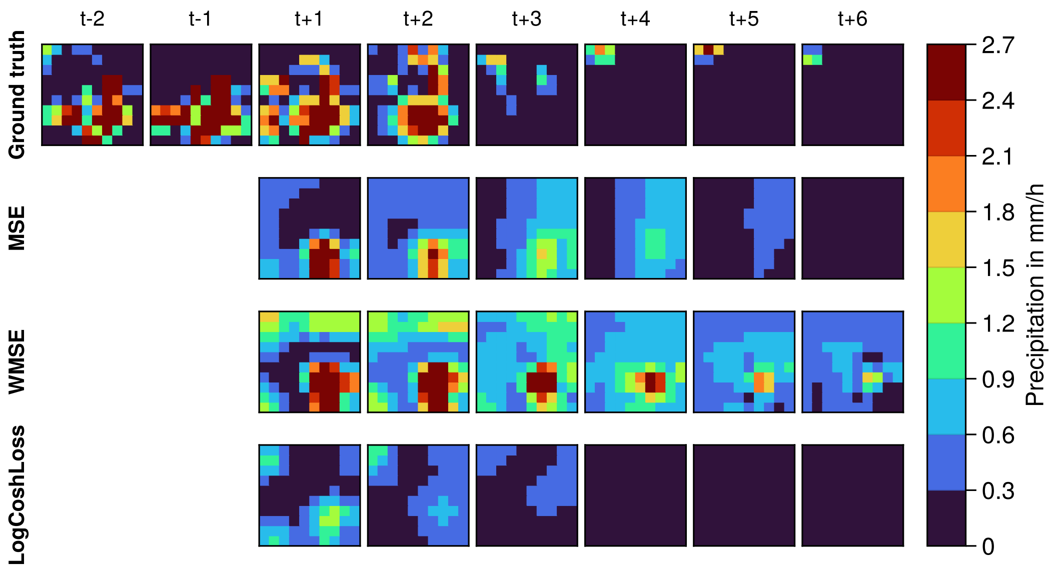

Figure 13: Model predictions compared to target in a 2.5° by 2.5° area. Shown are the outputs of the three different models at their best epochs with test data as input.

The different properties of the three different loss functions can be seen in figure 13, in which test data was used. All three models give somewhat plausible results, but with clear differences: MSE and WMSE give very similar predictions as a result of the second one being a modification of the first one. WMSE is, however, the only model that also predicts higher rainfall intensities in t+4 and t+5. Those are completely missing for the models MSE and Logcosh, but present in ERA5. These rain cells are locally misplaced for all three models.

Especially WMSE is poor at correctly predicting no rain, instead filling the entire image with blurred low-intensity rain. By contrast, LogCoshLoss shows lower gradients and underpredicts high intensities, but does a better job at representing areas without rain.

3.3 Forecast Skill

In section 2.6.1 we have described the fractions skill score (FSS) and its implementation. This score allows us to directly compare the prediction skill of different models at different spatial scales. We will compare the models using test data, show their behaviour on PGW data and finally test a presumably overfitted model.

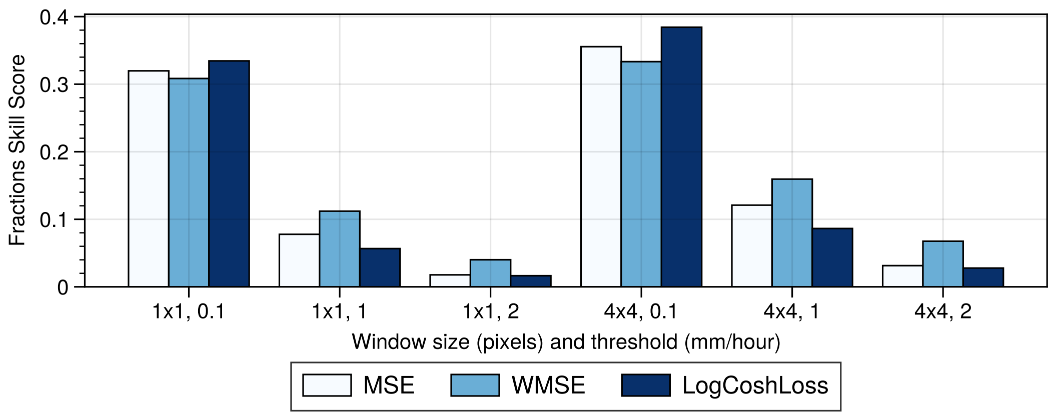

Figure 14: Mean FSS of the best model state of the three different models with the test dataset as input. The two window sizes equal 27.8 km $\times$ 27.8 km and 111.2 km $\times$ 111.2 km.

We have calculated the FSS for the unmodified test data to get a baseline for forecast performance, seen in figure 14, with the selection of window sizes (1×1 and 4×4 pixels) and thresholds ($0.1\,\mathrm{mm\,h^{-1}}$, $1\,\mathrm{mm\,h^{-1}}$ and $4\,\mathrm{mm\,h^{-1}}$) described in section 2.6.1.

First we observe the influence of the window size. For a given model and threshold, the score always increases when going from a 1×1 pixel window size to a 4×4 one. This is the exact expected result, as this window defines a tolerance in which the binary precipitation needs to match. We can observe one trend though: While for the smallest threshold this increase is very minor, for the largest thresholds all scores roughly double at the large window size. This means that forecasts are worse at predicting the precise location for precipitation exceeding $2\,\mathrm{mm\,h^{-1}}$, thus benefiting more strongly from an increased spatial tolerance.

Concerning the threshold, all models show a substantial loss of skill with an increasing threshold. This means that all models are worse at predicting higher intensities, which is a known problem with rainfall prediction using ML (Badrinath et al., 2023:297).

Finally, clear differences between the three loss functions can be observed. At a low threshold, LogCoshLoss shows the best performance, slightly better than MSE which itself is slightly better than WMSE. These differences are not large though. This changes with large thresholds, where WMSE strongly outperforms the other two loss functions.

There are several conclusions to be drawn from this: No loss functions of our selection is completely superior to another, instead having different strengths and weaknesses. LogCoshLoss shows the best performance in correctly predicting whether it rains in a pixel or not, but the worst performance at higher intensities. WMSE however is best at higher intensities, validating our weighted design.

So far, we have shown the result of our first sub-objective, creating a model for precipitation predictions. Now we will come to the main objective, the testing of their robustness when exposed to data modified with PGW. As described in section 2.5, we have used the test data for this. Previously, we calculated the FSS by comparing the prediction to the ground truth. Now we are comparing the prediction based on the modified data to the ground truth of the modified data. For plotting we selected three different levels of warming.

The results are shown in figure 15 for all three models. For the lowest threshold, all models show a slight loss of skill, which gets worse with increasing warming. This is to be expected, as this data is further away from the training domain (Freiesleben, Grote, 2023:109).

For a threshold of $1\,\mathrm{mm\,h^{-1}}$, this trend starts to change: All models show significant robustness, as there is no performance loss at a warming of $1\,\mathrm{K}$, with MSE even gaining a slight amount of skill. With further warming, the ability of MSE and LogCoshLoss degrades slightly, that of WMSE somewhat stronger than the other two.

At the highest threshold, the forecast skill of LogCoshLoss does not change significantly. For MSE und WMSE, however, it increases. This is an unexpected result, as the data is increasingly different from the training data under the assumption that training and test data share a similar underlying distribution. This will be further discussed in the next section.

In section 3.1 we assumed that our models reached a state of overfitting relatively early during training, as afterwards test loss increased. In order to test this we used a late epoch of the WMSE model, epoch 199, compared to the chosen best epoch 36. This figure 16 should thus be compared to the middle plot in figure 15. Contrary

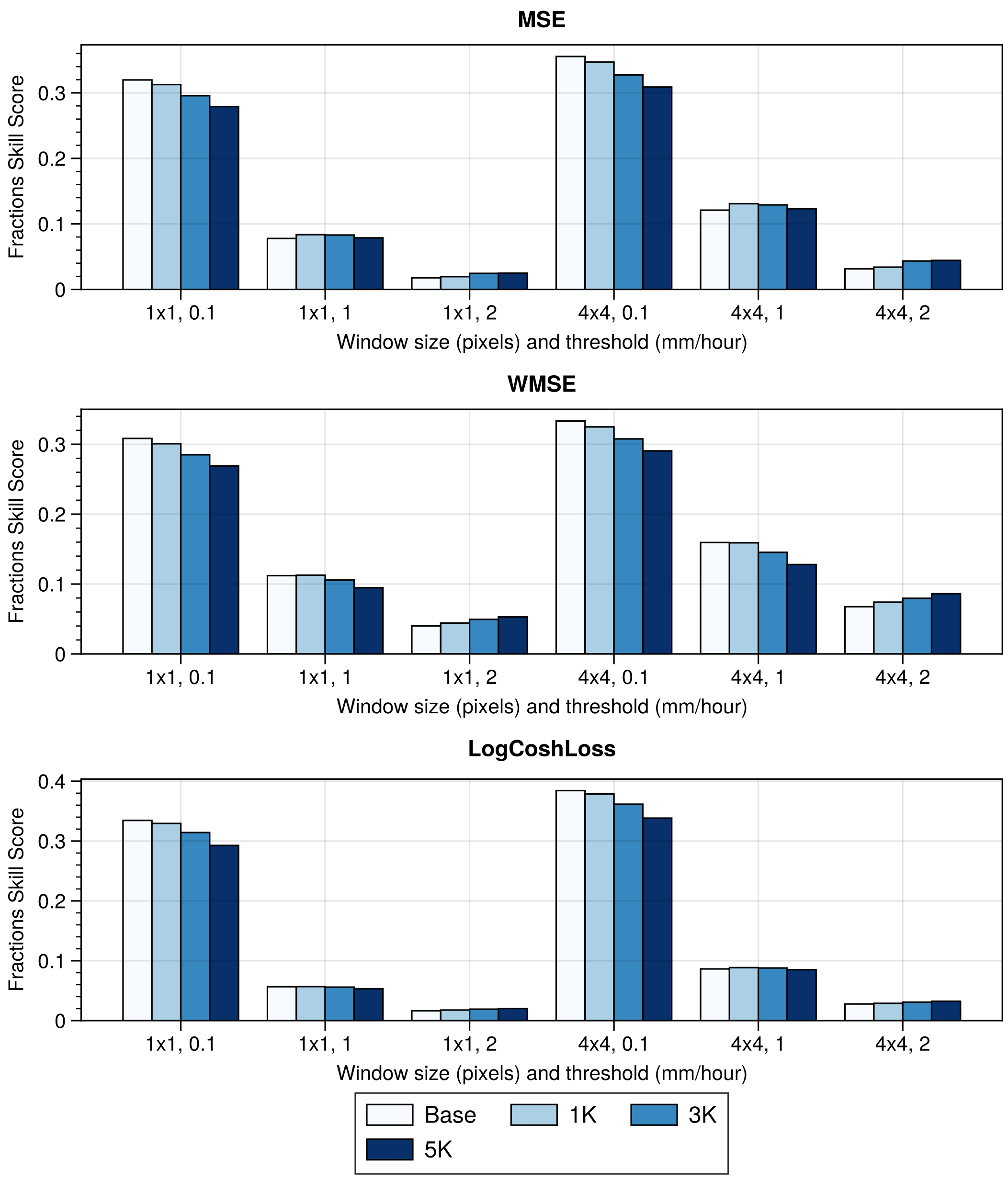

Figure 15: Mean FSS of the models using the three different loss functions with the test dataset and three different levels of warming at the respective epoch with the lowest loss

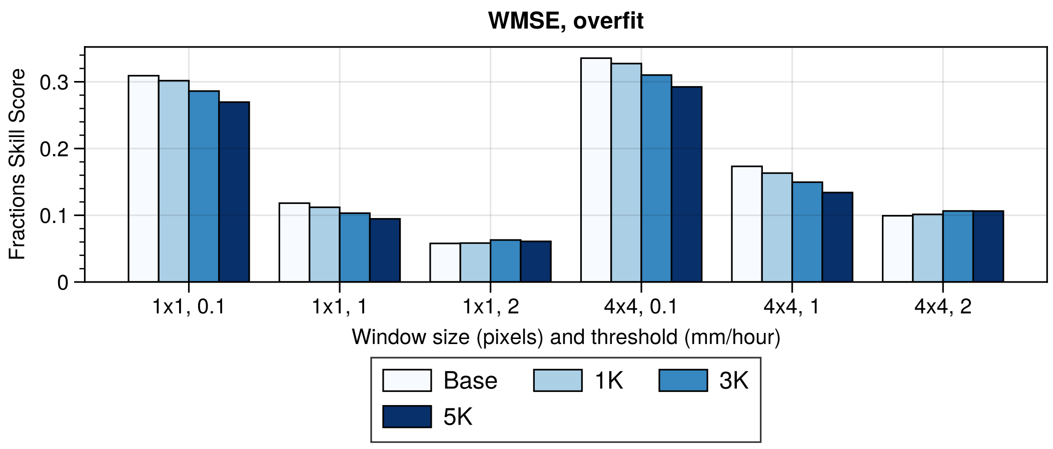

Figure 16: Mean FSS of the weighted MSE model with the test dataset and three different levels of warming at the final epoch **199**.

to the assumption, there is no degradation of performance, with this model state actually being slightly better at high precipitation intensities. This may indicate that the test loss is a poor indicator of finding the best model state in our case.

3.4 Spatial Plausibility of Predicted Rainfalls

In section 2.6.2 we described the implementation of the radial averaged power spectral density (RAPSD), which we will now calculate. This is a metric to calculate the spatial characteristics of rainfall.

Figure 17 shows the RAPSD for the ground truth, all models, PGW at 5 K warming and the prediction for PGW of the best performing model. The top plot shows the RAPSD without normalisation. This allows the direct comparison of precipitation sums between observation and prediction. First of all, all models significantly underestimate the total sum of rainfall, though this effect is worst for LogCoshLoss and least pronounced in WMSE. This plot indicates that our PGW does actually increase the total precipitation slightly. It is also of note that the underprediction of WMSE is nearly identical both without and with 5 K of warming.

The bottom plot in figure 17 is normalised and can thus be used to compare the power densities at different wavelengths, with the effect of the precipitation sum removed. Here, the differences between the models are marginal only. They all share one trait: At short wavelengths their underestimation is far stronger compared to the ground truth than at long wavelengths. This represents the blurring already shown in section 3.2: All models are relatively good at predicting large-scale rainfall patterns. They all perform significantly worse when the event has a very fine, highly varied structure, e.g. localised storm cells. This blurring is a known issue in ML

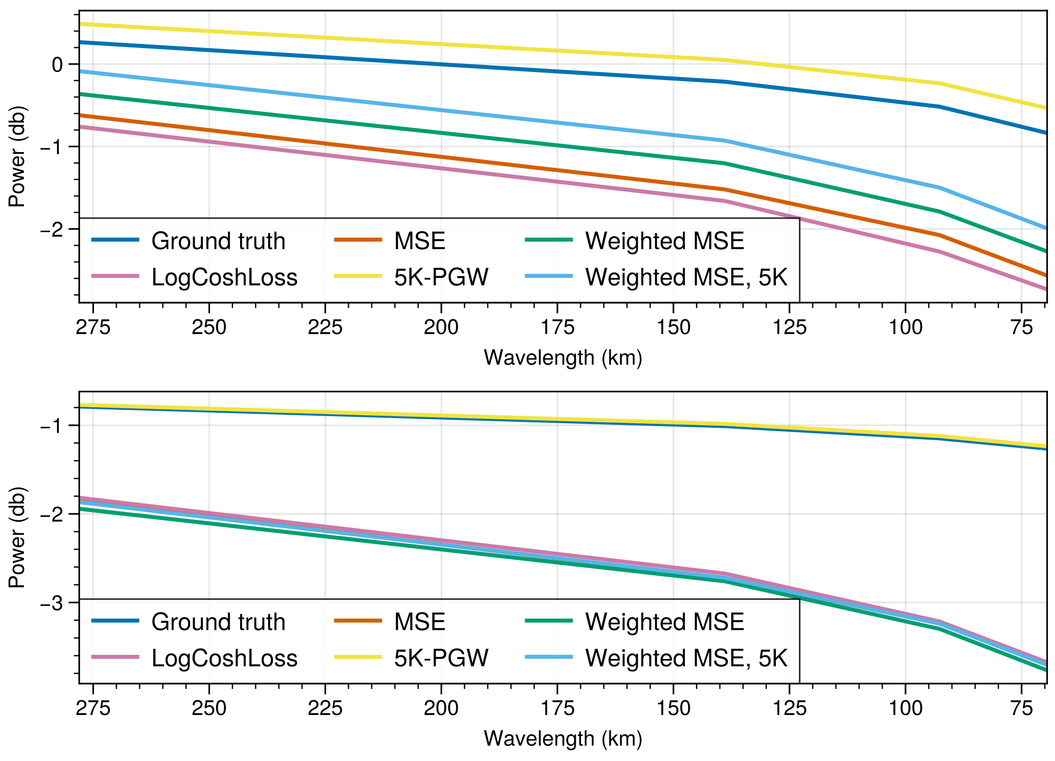

Figure 17: Mean RAPSD of the best model state of the three different models with the test dataset as input. The top plot is not normalised, showing the underestimation of precipitation. The bottom plot is normalised, which shows that all models are worse at small spatial scales. The RAPSD of the 5 K warmed data with one prediction is shown for comparison. Large wavelengths represent large rainfall patterns, low wavelengths represent finescale patterns.

precipitation forecasts (Harnist et al., 2024:3840).

Finally, we plotted the normalised predictions within PGW in figure 18. LogCoshLoss seems to be marginally more robust to changing data. Both plots look nearly identical to each other and to the normalised plot without warming, figure 17. This means that the capability of the model to produce low-scale structures in predictions remains mostly constant under PGW. In other words: The blurring does not get significantly better or worse.

4 Discussion

In the previous section 3 we presented the results of our analysis, as well as first likely explanations for the observations. In this section we will discuss how well our analysis manages to achieve the stated objectives, show limitations of our approach and present some possible alternative approaches.

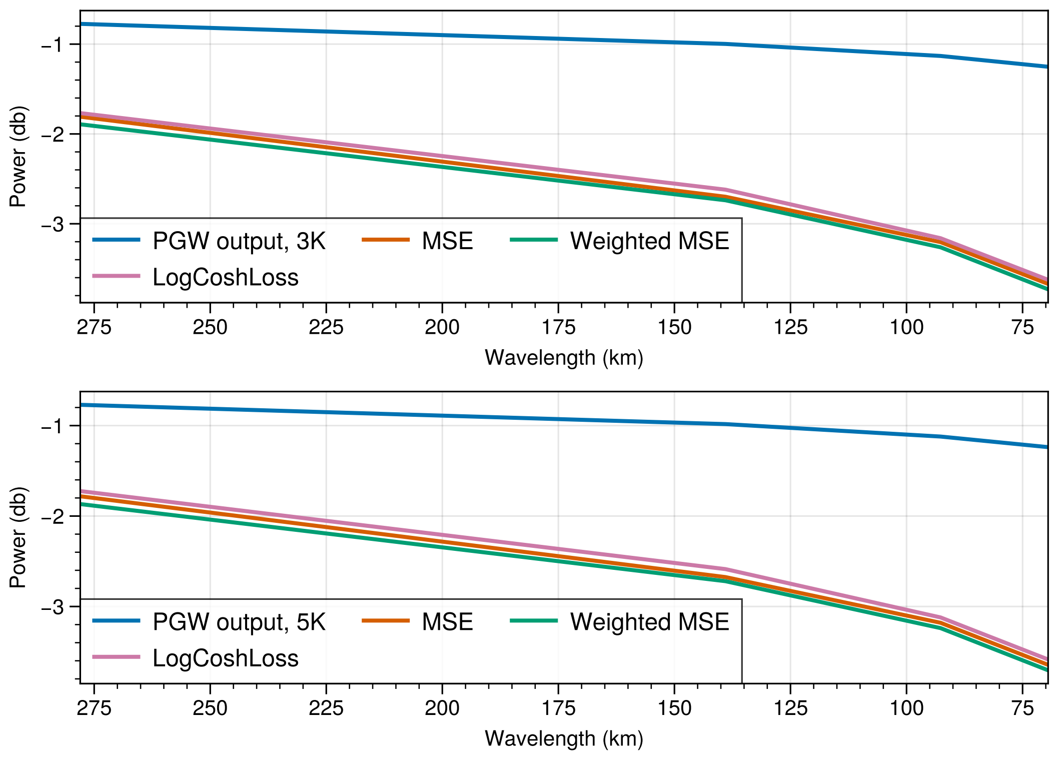

Figure 18: Comparison of the normalised mean RAPSD between the PGW output for 3 K (top) and 5 K (bottom) and predictions using that data as input.

4.1 Objective Completion

Our first sub-objective was the design of a ML short-term forecast model for precipitation, with specific focus on investigating different loss functions.

We have achieved this to a satisfactory degree. The qualitative evaluation shows that the model output is plausible. All FSS scores are good when the intensity is not considered, although far worse at a threshold of $2\,\mathrm{mm\,h^{-1}}$. The RAPSD of the predictions is close to the ground truth at high wavelengths, but it shows a significant drop in forecast performance at low wavelengths. In addition, we have managed to show differences in the prediction skill caused by the choice of loss function. WMSE shows by far the best performance at high rain intensities and does the least underprediction, while LogCoshLoss is best at predicting whether there is rain at all in an area or not, but has the largest total difference to the rain sum. One conclusion is that there is no model clearly superior to the others, instead the specific use case needs to be considered for that decision.

We do of course not achieve a similar performance to the state-of-the-art models such as AIFS and GraphCast. Considering that we use far simpler, computationally cheaper models with less input data, this is expected.

One weakness is the test loss, as shown in section 3.1. It reaches its optimum very quickly, somewhat increasing afterwards. Our assumption that this is due to overfitting seems flawed, as we show in section 3.3 and specifically figure 16 that a late epoch with higher test loss does not actually underperform compared to the seemingly optimal one. There are two ways to likely improve this in future research:

First of all, the very reason why we have chosen to use FSS and RAPSD as metrics instead of the loss is because our loss functions do not sufficiently take the properties of a good precipitation forecast into account. While we cannot use FSS as a loss functions due to the far higher computational complexity, we could calculate the FSS for all epochs and thus gain a better understanding of the performance.

Secondly, we skip the step of hyperparameter optimisation and do not have a separate validation dataset, which we did to simplify the analysis. In particular, this means that our learning rate is fixed during the whole training process, which possibly causes difficulties in finding better model weights.

Another problem is underestimation, an issue found with many models, such as Badrinath et al. (2023:302). It might be explained by all loss functions being applied pixelwise and being symmetric around 0. To clarify this, we imagine a fictional scenario in which all predictions are $0.1\,\mathrm{mm\,h^{-1}}$ below the ground truth and a second scenario in which half of the predictions each are this same amount below the ground truth and above it, respectively. The first scenario underpredicts the total rain amount, while the second one is precisely right, yet for both scenarios the total loss is identical. A second component to the loss functions which does not evaluate the individual pixel values, but rather e.g. the total sum of one hour, might significantly improve the adherence to the sum.

One of our loss functions, WMSE, contains parameters in the form of a threshold and a weight. These are little optimised so far and could potentially lead to a significantly stronger performance.

As previously discussed in section 2.2, the sample imbalance is a significant issue. High intensity precipitation events in particular are limited, which is known have an adverse effect on the forecast skill of a model (You et al., 2023:9). With the limits and sizes we use in our analysis, we only reject samples if each pixel contains less than $0.05\,\mathrm{mm\,h^{-1}}$ of hourly precipitation on average. A few pixels of high intensity rain are thus sufficient for the entire sample to be used, even if most of them contain no rain.

The fulfilment of the first objective, the creating of a model, allows for the second objective to be analysed: The testing of the model robustness when given data modified via PGW. More specifically, we do not just want the robustness in general, but also to see if there is a measurable difference between the loss functions under simulated warming. While we have gained insight into this question, our results do not conclusively answer it, which we will discuss.

We used two metrics to quantify the effect of PGW on our predictions, with different results. The RAPSD shows no noticable change for any given warming. The underprediction stays the same and the normalised curves are identical. This means that for our modelled climate change, the spatial structure of the model's outputs – in other words, the blurring – stays constant at different temperatures.

For low FSS thresholds, the score decreases similarly for all models with increasing warming. At high thresholds, however, the FSS increases slightly. This goes contrary to our expectations, as we would expect performance to decline the further from its training domain a model is used. Even the very skillful forecasts by Rackow et al. (2024) show a bias towards the conditions in which their models were trained.

One reason for this deviation from the expected result might be the possibly non-ideal selection of model states, which was already explained earlier in this section. We compared two WMSE models, at the epoch with the lowest loss and at an epoch at the end of training. We expected overfitting, but the latter model showed an improved FSS.

Another possible explanation lies in the PGW algorithm used, specifically the one for precipitation, which we described in section 2.5. To reiterate, here we use the FSS to compare the model output to the PGW-precipitation. Our algorithm reduces all precipitation below $2.1\,\mathrm{mm\,h^{-1}}$ and increases all above it. Because of the highly skewed distribution, this reduces the value for far more pixels than it increases, even though the mean stays the same. That threshold is very close to the FSS threshold used here of $2\,\mathrm{mm\,h^{-1}}$. Given the significant underestimation of all models which we showed in section 3.4, this might mean that for the majority of pixels, our PGW-algorithm actually counteracts this, leading to a better FSS.

As described in section 2.5, we use the Clausius-Clapeyron relation to directly increase precipitation by 7% per Kelvin of warming for large precipitation values. This is of course a big simplification. Although the total column water vapor (TCWV) and the total precipitation show a strong annual correlation in the tropics, their precise relationship is more complex (Trenberth, 2011:124). While using the Clausius-Clapeyron relation on the TCWV would likely be close prediction, using it on precipitation introduces further uncertainty.

The selected data only contains five variables. An additional one might improve the model performance and robustness, specifically TCWV. This variable is one of the inputs of models such as AIFS (Lang et al., 2024:4) and is also more directly affected by the Clausius-Clapeyron relation than precipitation.

In summary, our approach of using very simple, uniform data modifications to simulate climate change does not lead to clearly interpretable results, but this problem may be overcome with some modifications, or closer physical constraints.

4.2 Limits and Alternative Approaches

There are some fundamental limits caused by the design of our study, which we will elaborate.

Possibly the largest limitation of our analysis is the limited direct comparability of our results. We have observed effects such as the robustness of the spatial characteristics, but we do not know how our model would react when given more realistic PGW data.

All of our data was deliberately selected from the same area, which removed potentially differing data quality as a variable in our analysis. This does mean, however, that we have no information about how capable our models are at spatial generalisation, that is being trained with data from one area and used with data from a different area.

The training data is generated from a total of four years (1980, 1981, 2020 and 2021) while the test data is from 2000 and 2001. These relatively low numbers might cause a significant impact of inter-annual variability, as some of those years might be unusually dry or wet. This could easily be improved for a study of larger scope by downloading more data.

Regarding our simple implementation of PGW we have chosen to directly increase the temperature. This simplified the modelling process, but means that we are unable to draw conclusions to the temperature behaviour of our models in a warmed climate. By comparison, Rackow et al. (2024:6-7) have shown a cold bias over land in all of their models, though to different extents in different models.

The purpose of our PGW-approach is to create a reasonable change in the data with which we analyse the robustness of the model. As such, the scaling equation 2 for precipitation in section 2.5 is built to keep to several constraints, while providing well-behaved data. This is by no means the most realistic prediction possible, as it does not account for the complex effects of the Hadley circulation and especially large seasonal differences (Cook et al., 2025:381). In addition, we only apply this pixelwise, so possible changes in precipitation structure are not modelled.

Apart from these limitations, there are also alternative approaches that could lead to better results. Badrinath et al. (2023:294) have, just like us, encountered the problem of underestimation in their modelling and propose a modification to the loss function – MSE in their case – to remedy this. They add asymmetry to the function in form of a weight w_s, which is a hyperparameter that they optimise on the validation dataset. Using the following equation for loss, they use the unmodified MSE for a prediction exceeding the ground truth, but multiply it by the weight for underpredictions.

$$l(y_{\text{pred}}, y_{\text{true}}) = \begin{cases} y_{\text{pred}}, y_{\text{true}})^2 & y_{\text{pred}} \geq y_{\text{true}} \\ w_s(y_{\text{pred}}, y_{\text{true}})^2 & else \end{cases} \tag{10}$$

This approach does not fully remove the problem of underestimation, but significantly improves it (Badrinath et al., 2023:304). It is likely that a similar addition to our loss functions would also lead to a higher prediction skill.

There are further ways to implement weights to the loss function. Lang et al. (2024:5) use MSE for their AIFS, but with weights for area and height as well as loss scaling, so that all variables contribute roughly equally. Furthermore, different loss functions such as Huber Loss could be tested.

We perform the prediction of six hours in a single model pass. An alternative exists in autoregressive prediction, in which the first prediction step is then used to again predict the step afterwards (Ayzel et al., 2020:2641). This has the property of smoothing predictions, though, and is thus especially bad at predicting high-intensity events (Ayzel et al., 2020:2641). That means our decision to directly predict everything is likely the more appropriate choice.

One technique we did not use in our model architecture is Dropout, even though it might benefit the robustness of the model. It was introduced by Srivastava et al. (2014) to address the problem of overfitting, in which a model learns the pattern of the training data very well, but then has problems with previously unseen data. They mitigated this by creating several sub-networks of which a percentage of weights was dropped. This considerably improves performance, but also increases training time by a factor of two to three (Srivastava et al., 2014:1952). As such our choice to not use it is likely reasonable given the technical constraints, but its use should be considered in future research.

A further way to potentially improve the robustness of this CNN is presented by Li et al. (2020), who specifically target the 2-D convolutional layer as a source of redundancy. This causes an inefficient parameter size (Li et al., 2020:2). They propose adding a learnable sparse transform (LST) to the model, which transforms the data in order to reduce different forms of redundancy. The LST learns this transformation during training and does not need manual design (Li et al., 2020:2). Li et al. (2020:4) also add hybrid soft thresholding to the basic activation function ReLU, finding that these changes improve the robustness of the model to changing data. Considering that they directly compare their development to a ResNet – which is the type of model we use – and show improved results, our model would likely benefit from the addition of an LST as well.

5 Conclusion

The goal of our analysis was finding out how the performance of a ML model for precipitation forecasts changes when the input data is modified via a simple PGW-algorithm. To achieve this we first downloaded ERA5-data and processed it into data samples. We used them to train three versions of a ResNet neural network with a different loss function each, MSE, WMSE and LogCoshLoss. We also implemented a simple PGW-algorithm. To test the performance of the models for the unmodified and the PGW-data we used two different metrics.

We have managed to train models with good precipitation forecast skill for the size of our study. All models show comparable spatial characteristics, blurring out small features. The model which uses LogCoshLoss is the best at forecasting if an area will experience rain in general, while the WMSE performs better than the other loss functions for higher intensities of precipitation. All models significantly underpredict total rain sums, though this is worst for LogCoshLoss. We conclude that the choice of loss function depends on the specific use case.

Our study is not fully capable of assessing to a conclusive degree the robustness of our precipitation forecast under climate change: The capability of two models actually increases for increased warming and high precipitation intensities, which goes against well-established principles of out-of-distribution predictions, such as described by Rackow et al. (2024). This suggests that a PGW with more physical constraints is needed.

We have gained one significant insight on the topic of robustness to climate change: The spatial characteristics of the prediction do not significantly change with our implementation of PGW; the models do not compensate predictions further outside of the training domain with increased blurring.

There is still very little research on using ML to predict future climates. The work of Rackow et al. (2024) has already been presented, but their approach still requires large computational resources. A middle ground might be a worthwhile research avenue, with an approach more sophisticated than ours, but simpler than global, state-of-the-art models. This means be the PGW4ERA5 package by Brogli et al. (2023) could be a logical step to continue this research. Given the immense pace of progress in the field of climate science and machine learning, new innovations will near certainly be available for future research, possibly being able to inform real-world decisions in response to climate change.

Bibliography

Ayzel G., Heistermann M., Winterrath T. (2019): Optical flow models as an open benchmark for radar-based precipitation nowcasting (rainymotion v0.1). In: Geoscientific Model Development, 12(4), S. 1387–1402. DOI: 10.5194/gmd-12-1387-2019.
`https://gmd.copernicus.org/articles/12/1387/2019/` (11.03.2025).

Ayzel G., Scheffer T., Heistermann M. (2020): RainNet v1.0: a convolutional neural network for radar-based precipitation nowcasting. In: Geoscientific Model Development, 13(6), S. 2631–2644. DOI: 10.5194/gmd-13-2631-2020.

Badrinath A., Delle Monache L., Hayatbini N., Chapman W., Cannon F., Ralph M. (2023): Improving precipitation forecasts with convolutional neural networks. In: Weather and Forecasting, 38(2), S. 291–306. ISSN 0882-8156. DOI: 10.1175/WAF-D-22-0002.1.

Ban N., Caillaud C., Coppola E., Pichelli E., Sobolowski S., Adinolfi M., Ahrens B., Alias A., Anders I., Bastin S., Belušić D., Berthou S., Brisson E., Cardoso R. M., Chan S. C., Christensen O. B., Fernández J., Fita L., Frisius T., Gašparac G., Giorgi F., Goergen K., Haugen J. E., Hodnebrog Ø., Kartsios S., Katragkou E., Kendon E. J., Keuler K., Lavin-Gullon A., Lenderink G., Leutwyler D., Lorenz T., Maraun D., Mercogliano P., Milovac J., Panitz H.-J., Raffa M., Remedio A. R., Schär C., Soares P. M. M., Srnec L., Steensen B. M., Stocchi P., Tölle M. H., Truhetz H., Vergara-Temprado J., de Vries H., Warrach-Sagi K., Wulfmeyer V., Zander M. J. (2021): The first multi-model ensemble of regional climate simulations at kilometer-scale resolution, part I: evaluation of precipitation. In: Climate Dynamics, 57(1), S. 275–302. ISSN 1432-0894. DOI: 10.1007/s00382-021-05708-w.

Ben Bouallègue Z., Clare M. C. A., Magnusson L., Gascón E., Maier-Gerber M., Janoušek M., Rodwell M., Pinault F., Dramsch J. S., Lang S. T. K., Raoult B., Rabier F., Chevallier M., Sandu I., Dueben P., Chantry M., Pappenberger F. (2024): The rise of data-driven weather forecasting: A first statistical assessment of machine learning–based weather forecasts in an operational-like context. In: Bulletin of the American Meteorological Society, 105(6), S. E864–E883. ISSN 0003-0007. DOI: 10.1175/BAMS-D-23-0162.1.

Brogli R., Heim C., Mensch J., Sørland S. L., Schär C. (2023): The pseudo-global-warming (PGW) approach: methodology, software package PGW4ERA5 v1.1, validation, and sensitivity analyses. In: Geoscientific Model Development, 16(3), S. 907–926. DOI: 10.5194/gmd-16-907-2023.

Cook K. H., Vizy E. K., Andrews P. C., Zhao S. (2025): Temperature increases threaten the Congo basin rainforest in the twenty-first century more than precipitation changes. In: Journal of Climate, 38(1), S. 369–386. ISSN 0894-8755. DOI: 10.1175/JCLI-D-24-0289.1.

ECMWF: Copernicus Knowledge Base. (2023). `https://confluence.ecmwf.int/display/CKB/ERA5%3A+What+is+the+spatial+reference` (04.03.2025).

ECMWF: Parameter Database. (2025). `https://codes.ecmwf.int/grib/param-db/` (04.03.2025).

Freiesleben T., Grote T. (2023): Beyond generalization: a theory of robustness in machine learning. In: Synthese, 202(4). DOI: 10.1007/s11229-023-04334-9.

Gheysari A. F., Maghoul P., Ojo E. R., Shalaby A. (2023): Reliability of ERA5 and ERA5-Land reanalysis data in the Canadian Prairies. In: Theoretical and Applied Climatology, 155(4), S. 3087–3098. ISSN 0177-798X. DOI: 10.1007/s00704-023-04771-z.

Gibson P. B., Chapman W. E., Altinok A., Delle Monache L., DeFlorio M. J., Waliser D. E. (2021): Training machine learning models on climate model output yields skillful interpretable seasonal precipitation forecasts. In: Communications Earth & Environment, 2(1). DOI: 10.1038/s43247-021-00225-4.

Gudivada V. N., Apon A., Ding J. (2017): Data quality considerations for big data and machine learning: Going beyond data cleaning and transformation. In: International Journal on Advances in Software, 10(1 & 2).

Harnist B., Pulkkinen S., Mäkinen T. (2024): Deuce v1.0: a neural network for probabilistic precipitation nowcasting with aleatoric and epistemic uncertainties. In: Geoscientific Model Development, 17(9), S. 3839–3866. DOI: 10.5194/gmd-17-3839-2024.
`https://gmd.copernicus.org/articles/17/3839/2024/` (05.03.2025).

He K., Zhang X., Ren S., Sun J.: Deep Residual Learning for Image Recognition. (2015).
`http://arxiv.org/pdf/1512.03385v1` (21.02.2025).

Heim C., Leutwyler D., Schär C. (2023): Application of the Pseudo–Global Warming approach in a kilometer–resolution climate simulation of the tropics. In: Journal of Geophysical Research: Atmospheres, 128(5). ISSN 2169-897X. DOI: 10.1029/2022JD037958.

Hersbach H., Bell B., Berrisford P., Hirahara S., Horányi A., Muñoz-Sabater J., Nicolas J., Peubey C., Radu R., Schepers D., Simmons A., Soci C., Abdalla S., Abellan X., Balsamo G., Bechtold P., Biavati G., Bidlot J., Bonavita M., de Chiara G., Dahlgren P., Dee D., Diamantakis M., Dragani R., Flemming J., Forbes R., Fuentes M., Geer A., Haimberger L., Healy S., Hogan R. J., Hólm E., Janisková M., Keeley S., Laloyaux P., Lopez P., Lupu C., Radnoti G., de Rosnay P., Rozum I., Vamborg F., Villaume S., Thépaut J.-N. (2020): The ERA5 global reanalysis. In: Quarterly Journal of the Royal Meteorological Society, 146(730), S. 1999–2049. ISSN 0035-9009. DOI: 10.1002/qj.3803.

Ioffe S., Szegedy C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Bach F., Blei D. [Editors]: Proceedings of the 32nd International Conference on Machine Learning. PMLR, Lille, France, (2015), 37von Proceedings of Machine Learning Research, S. 448–456. `https://proceedings.mlr.press/v37/ioffe15.html` (10.03.2025).

IPCC (2023): Climate Change 2023: Synthesis Report. Contribution of Working Groups I, II and III to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change. IPCC, Geneva, Switzerland. DOI: 10.59327/IPCC/AR6-9789291691647.

Johnson J. M., Khoshgoftaar T. M. (2019): Survey on deep learning with class imbalance. In: Journal of Big Data, 6(1), S. 27. ISSN 2196-1115. DOI: 10.1186/s40537-019-0192-5.

Jung C., Schindler D. (2019): Changing wind speed distributions under future global climate. In: Energy Conversion and Management, 198, S. 111841. ISSN 01968904. DOI: 10.1016/j.enconman.2019.111841.

Kingma D. P., Ba J.: Adam: A method for stochastic optimization. In: Bengio Y., LeCun Y. [Editors]: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. (2015).
`http://arxiv.org/abs/1412.6980`.

Lam R., Sanchez-Gonzalez A., Willson M., Wirnsberger P., Fortunato M., Alet F., Ravuri S., Ewalds T., Eaton-Rosen Z., Hu W., Merose A., Hoyer S., Holland G., Vinyals O., Stott J., Pritzel A., Mohamed S., Battaglia P. (2023): Learning skillful medium-range global weather forecasting. In: Science, 382(6677), S. 1416–1421. DOI: 10.1126/science.adi2336. `https://www.science.org/doi/pdf/10.1126/science.adi2336`,
`https://www.science.org/doi/abs/10.1126/science.adi2336`.

Lang S., Alexe M., Chantry M., Dramsch J., Pinault F., Raoult B., Clare M. C. A., Lessig C., Maier-Gerber M., Magnusson L., Bouallègue Z. B., Nemesio A. P., Dueben P. D., Brown A., Pappenberger F., Rabier F.: Aifs – ecmwf's data-driven forecasting system. (2024). 2406.01465, https://arxiv.org/abs/2406.01465 (11.03.2025).

Lavers D. A., Simmons A., Vamborg F., Rodwell M. J. (2022): An evaluation of ERA5 precipitation for climate monitoring. In: Quarterly Journal of the Royal Meteorological Society, 148(748), S. 3152–3165. ISSN 0035-9009. DOI: 10.1002/qj.4351.

Leeuwenburg T., Loveday N., Ebert E. E., Cook H., Khanarmuei M., Taggart R. J., Ramanathan N., Carroll M., Chong S., Griffiths A., Sharples J. (2024): scores: A Python package for verifying and evaluating models and predictions with xarray. In: Journal of Open Source Software, 9(99), S. 6889. DOI: 10.21105/joss.06889.

Li L., Wang K., Li S., Feng X., Zhang L.: Lst-net: Learning a convolutional neural network with a learnable sparse transform. In: Vedaldi A., Brox H. B. T., Frahm J.-M. [Editors]: Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X. Springer-Verlag, Berlin, Heidelberg, (2020), S. 562–579. ISBN 978-3-030-58606-5. DOI: 10.1007/978-3-030-58607-2_33 . https://www.ecva.net/papers/eccv_2020/papers_ECCV/papers/123550562.pdf (07.03.2025).

Magar R., Barati Farimani A. (2023): Learning from mistakes: Sampling strategies to efficiently train machine learning models for material property prediction. In: Computational Materials Science, 224, S. 112167. ISSN 09270256. DOI: 10.1016/j.commatsci.2023.112167.

NCEI: Monthly climate reports, annual 2023. (2023). https://www.ncei.noaa.gov/access/monitoring/monthly-report/global/202313 (01.03.2025).

Necker T., Wolfgruber L., Kugler L., Weissmann M., Dorninger M., Serafin S. (2024): The fractions skill score for ensemble forecast verification. In: Quarterly Journal of the Royal Meteorological Society, 150(764), S. 4457–4477. ISSN 0035-9009. DOI: 10.1002/qj.4824.

Olivetti L., Messori G. (2024): Do data-driven models beat numerical models in forecasting weather extremes? a comparison of ifs hres, pangu-weather, and graphcast. In: Geoscientific Model Development, 17(21), S. 7915–7962. DOI: 10.5194/gmd-17-7915-2024. https://gmd.copernicus.org/articles/17/7915/2024/.

O'Shea K., Nash R.: An introduction to convolutional neural networks. (2015). `1511.08458`, `https://arxiv.org/abs/1511.08458` (10.03.2025).

Pielke Jr R., Burgess M. G., Ritchie J. (2022): Plausible 2005–2050 emissions scenarios project between 2 °C and 3 °C of warming by 2100. In: Environmental Research Letters, 17(2), S. 024027. DOI: 10.1088/1748-9326/ac4ebf.

Polz J., Glawion L., Gebisso H., Altenstrasser L., Graf M., Kunstmann H., Vogl S., Chwala C. (2024): Temporal super-resolution, ground adjustment, and advection correction of radar rainfall using 3-D-convolutional neural networks. In: IEEE Transactions on Geoscience and Remote Sensing, 62, S. 1–10. ISSN 0196-2892. DOI: 10.1109/TGRS.2024.3371577.

Pulkkinen S., Nerini D., Pérez Hortal A. A., Velasco-Forero C., Seed A., Germann U., Foresti L. (2019): Pysteps: an open-source python library for probabilistic precipitation nowcasting (v1.0). In: Geoscientific Model Development, 12(10), S. 4185–4219. DOI: 10.5194/gmd-12-4185-2019. `https://gmd.copernicus.org/articles/12/4185/2019/` (05.03.2025).

Rackow T., Koldunov N., Lessig C., Sandu I., Alexe M., Chantry M., Clare M., Dramsch J., Pappenberger F., Pedruzo-Bagazgoitia X., Tietsche S., Jung T.: Robustness of AI-based weather forecasts in a changing climate. (2024). `http://arxiv.org/pdf/2409.18529v1` (05.03.2025).

Rivoire P., Martius O., Naveau P. (2021): A comparison of moderate and extreme ERA–5 daily precipitation with two observational data sets. In: Earth and Space Science, 8(4). ISSN 2333-5084. DOI: 10.1029/2020EA001633.

Roberts N. M., Lean H. W. (2008): Scale-selective verification of rainfall accumulations from high-resolution forecasts of convective events. In: Monthly Weather Review, 136(1), S. 78 – 97. DOI: 10.1175/2007MWR2123.1. `https://journals.ametsoc.org/view/journals/mwre/136/1/2007mwr2123.1.xml` (04.03.2025).

Ruzanski E., Chandrasekar V. (2011): Scale filtering for improved nowcasting performance in a high-resolution X-band radar network. In: IEEE Transactions on Geoscience and Remote Sensing, 49(6), S. 2296–2307. ISSN 0196-2892. DOI: 10.1109/TGRS.2010.2103946.

Sai Y., Jinxia R., Zhongxia L.: Learning of neural networks based on weighted mean squares error function. In: 2009 Second International Symposium on Computational Intelligence and Design. (2009), 1, S. 241–244. DOI: 10.1109/ISCID.2009.67.

Saleh R. A., Saleh A. K. M. E.: Statistical properties of the log-cosh loss function used in machine learning. (2024). 2208.04564, https://arxiv.org/abs/2208.04564 (04.03.2025).

Schmidt D. F., Grise K. M. (2017): The Response of Local Precipitation and Sea Level Pressure to Hadley Cell Expansion. In: Geophysical Research Letters, 44(20). ISSN 0094-8276. DOI: 10.1002/2017GL075380.

Schär C., Fuhrer O., Arteaga A., Ban N., Charpilloz C., Girolamo S. D., Hentgen L., Hoefler T., Lapillonne X., Leutwyler D., Osterried K., Panosetti D., Rüdisühli S., Schlemmer L., Schulthess T. C., Sprenger M., Ubbiali S., Wernli H. (2020): Kilometer-scale climate models: Prospects and challenges. In: Bulletin of the American Meteorological Society, 101(5), S. E567 – E587. DOI: 10.1175/BAMS-D-18-0167.1. https://journals.ametsoc.org/view/journals/bams/101/5/bams-d-18-0167.1.xml (10.03.2025).

Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R. (2014): Dropout: A simple way to prevent neural networks from overfitting. In: Journal of Machine Learning Research, 15, S. 1929–1958.

Tian Y., Su D., Lauria S., Liu X. (2022): Recent advances on loss functions in deep learning for computer vision. In: Neurocomputing, 497, S. 129–158. ISSN 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2022.04.127. https://www.sciencedirect.com/science/article/pii/S0925231222005239 (07.03.2025).

Trenberth K. E. (2011): Changes in precipitation with climate change. In: Climate Research, 47(1), S. 123–138. DOI: 10.3354/cr00953.

Wan X. (2019): Influence of feature scaling on convergence of gradient iterative algorithm. In: Journal of Physics: Conference Series, 1213(3), S. 032021. ISSN 1742-6588. DOI: 10.1088/1742-6596/1213/3/032021.

Weerts H. J. P., Mueller A. C., Vanschoren J.: Importance of tuning hyperparameters of machine learning algorithms. (2020). 2007.07588, https://arxiv.org/abs/2007.07588 (10.03.2025).

You X.-x., Liang Z.-m., Wang Y.-q., Zhang H. (2023): A study on loss function against data imbalance in deep learning correction of precipitation forecasts. In: Atmospheric Research, 281, S. 106500. ISSN 01698095. DOI: 10.1016/j.atmosres.2022.106500.

Appendix

On `https://github.com/alexander-mayer/bachelor-thesis` the source code is made available. A PDF file of this thesis can be found there as well for ease of reading with some personal information redacted.