FernUniversität in Hagen
Faculty of Mathematics and Computer Science

AIG
Artificial Intelligence Group

DWD
Deutscher Wetter-dienst

PIK
Potsdam Institute for Climate Impact Research

Machine Learning Techniques for Inpainting of Climate Data: A Comparative Study

Master's Thesis

in partial fulfilment of the requirements for
the degree of *Master of Science (M.Sc.)*
in Data Science

submitted by
Merlin Jo Hosak

First examiner: Prof. Dr. Matthias Thimm
Artificial Intelligence Group

Advisor: Dr. Philipp Heß
Potsdam Institute for Climate Impact Research

Statement

I declare that I have written the master´s thesis independently and without unauthorized use of third parties. I have only used the indicated resources and I have clearly marked the passages taken verbatim or in the sense of these resources as such. The assurance of independent work also applies to any drawings, sketches or graphical representations. The work has not previously been submitted in the same or similar form to the same or another examination authority and has not been published. By submitting the electronic version of the final version of the master´s thesis, I acknowledge that it will be checked by a plagiarism detection service to check for plagiarism and that it will be stored exclusively for examination purposes.

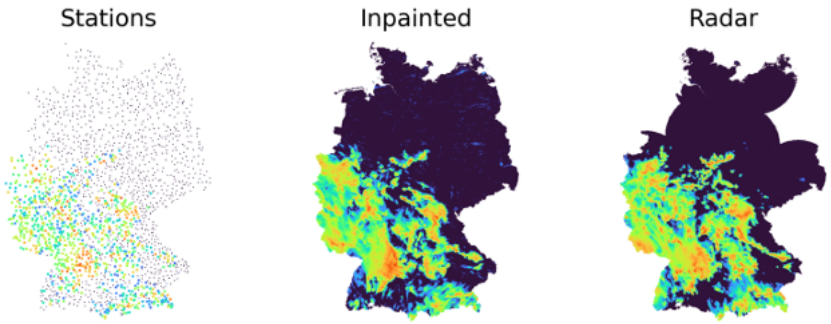
I explicitly agree to have this thesis published on the webpage of the artificial intelligence group and endorse its public availability.

Software created for this work has been made available as open source; a corresponding link to the sources is included in this work. The same applies to any research data.

Balangoda, 6th of February 2026 .....

(Place, Date) (Signature)

Graphical Abstract



Abstract

Due to climate change, the frequency of extreme weather events is increasing. High resolution and quality precipitation field data is necessary for accurate weather forecasts, but the available historic data is limited to specific stations and timestamps. Precipitation data is right skewed, highly localized and non Gaussian, making it hard to estimate precipitation between measurement stations.

In Germany, the statistical REGNIE method is currently used by the German Weather Service for this purpose. This thesis compares REGNIE to two machine learning methods: a newly implemented denoising diffusion probabilistic model (DDPM), and a U-Net based inpainting approach from another investigation.

Whilst the statistical method performs better than the machine learning models, an improved version of the DDPM performs comparably to REGNIE. This study suggests several improvements, including the use of more spatio-temporal context information as well as the use of LSTM cells in the architecture.

It outlines the potential of these models, particularly DDPMs, for use in climate data estimation as well as their limitations. The work consists of an in-depth model comparison for inpainting climate data and provides a basis for future AI method comparison in this field.

Contents

1. Introduction **1**
- 1.1. Motivation . . . 1
- 1.2. Image Inpainting . . . 1
- 1.3. Related Work . . . 2
- 1.4. Preliminaries . . . 2
  - 1.4.1. REGNIE . . . 2
  - 1.4.2. U-Net . . . 4
  - 1.4.3. Denoising Diffusion Probabilistic Models . . . 7
- 1.5. Aim . . . 11

2. Methods **11**
- 2.1. Data . . . 11
  - 2.1.1. Sources . . . 11
  - 2.1.2. Data Preparation . . . 13
- 2.2. Model Design . . . 19
  - 2.2.1. Architecture . . . 20
  - 2.2.2. Hyperparameter Optimization . . . 21
  - 2.2.3. Sampling and Inpainting . . . 23
  - 2.2.4. Evaluation . . . 26
  - 2.2.5. Over- & Underfitting . . . 28

3. Experiment I: Hourly Model **29**
- 3.1. Training . . . 29
- 3.2. Results . . . 31
  - 3.2.1. Inpainting . . . 31
  - 3.2.2. Metrics . . . 33
- 3.3. Experiment Discussion . . . 35

4. Experiment II: Daily Model - HYRAS comparison **37**
- 4.1. Training . . . 37
- 4.2. Results . . . 37
  - 4.2.1. Inpainting . . . 40
  - 4.2.2. Metrics . . . 40
- 4.3. Experiment Discussion . . . 44

5. Experiment III: Sparse Hourly Model - U-Net comparison **46**
- 5.1. Filippou Comparison . . . 47
- 5.2. Results . . . 48
  - 5.2.1. General . . . 48
  - 5.2.2. U-Net comparison . . . 50
  - 5.2.3. HYRAS comparison . . . 53
- 5.3. Experiment Discussion . . . 55

6. Discussion **56**
6.1. Improvements . . . 56
6.1.1. Data . . . 57
6.1.2. U-Net . . . 58
6.1.3. Training . . . 59
6.1.4. Sampling . . . 59
6.1.5. General Setup . . . 60
6.1.6. Ranking . . . 61
6.2. Ideas . . . 63
6.3. Interpretation . . . 64
6.3.1. Contribution . . . 64
6.3.2. Field Context . . . 66
6.3.3. Next steps . . . 67

7. Conclusion **68**

A. Code **69**

B. U-Net Model Investigation **69**

Acknowledgments

It takes a village to write a thesis.

First and foremost, I would like to thank Dr. Philipp Heß from the Potsdam Institute of Climate Impact Research (PIK) for the guidance in the field of climate data and image inpainting over the last months, as well as initiating this project. I have learned so much in this project and am really thankful for the in-depth explanations, motivation, kind words, and bearing with me in general.

Next, I would like to thank Prof. Dr. Matthias Thimm for allowing me to do a rather complicated thesis project across three institutions. I am very happy I got to work on a relevant topic and hopefully could make a small scientific contribution in a field that I care about.

I would like to thank Svenja Hoffmann and Dr. Markus Drüke from the Deutsche Wetterdienst (DWD) for helping me understand the context of the work, providing valuable resources and always engaging with my questions. Its been very exciting for me to get a peak into your work.

Further, I would like to thank Johannes Meuer from the Deutsche Klimarechenzentrum (DKRZ), Danai Filippou from the Max Planck Institute für Meteorologie (MPI) and Christian Burmester for answering my inpainting inquiries.

On a personal note, I would like to thank Johanna Murray and Jakob Harteg for helping me with the final steps. It's very fulfilling to have such capable and kind friends around.

My housemates at Bergauf have been cooking me food, helping me with my chores, and motivating me after sleepless nights. Vielen Dank!

Of course, I would like to thank my parents, for believing in me and letting me be, but still supporting me all the same. I am considering myself very lucky to have you in my life.

Lastly, I want to thank Iso - this thesis has been a bit of a pain to you, I guess, and I am deeply grateful (and astonished) how you still supported me all the way through immensely, in all kinds of ways. It's the utmost joy to be with you.

On the Use of AI Tools

In this investigation, the artificial intelligence (AI) tools ChatGPT and Github Copilot were used.

Specifically, the tools were used for literature comprehension, interpretation and suggestions as well as code inspiration, the compiling of LaTeX tables and algorithms and code autocompletion. It was not used for text generation or checking.

All AI information used in the thesis has been verified and cross referenced with other sources by the author. Therefore, the responsibility for all claims and results in the thesis lies with the author.

List of Figures

1. U-Net architecture from the original Ronneberger et al. paper [48]. . 5
2. U-Net Architecture used by Filippou et al. [13] 8
3. DDPM forward and backward diffusion process (from [20]). 9
4. Magnitude and density of precipitation stations in Germany from [39] 12
5. Location and range of the DWD radars 14
6. Various factors can decrease radar quality (from [23], see [13]) 15
7. Distances between HYRAS grid coordinates and the assigned *RADKLIM* grid coordinates 16
8. Radar precipitation data transformed with *Asinh*, Log & *Box-Cox* standardizers. 20
9. The Diffusion Posterior Sampling inpainting process 25
10. Training of the hourly model over 250 epochs, including training and validation loss. 30
11. Samples produced by the hourly model at different epochs 31
12. Hourly Model Histograms and RAPSD curves 32
13. Test of different lambda values in inpainting 33
14. Inpainted hourly station data at six random time steps 34
15. Hourly model results over time 35
16. Training of the daily model - significant improvement of validation loss seems to stop after epoch 181. 38
17. Samples produced by the daily model at different epochs compared to real samples from the ground truth RADKLIM dataset. 38
18. Histograms and RAPSD plots of daily model samples at different epochs 39
19. Daily Model vs HYRAS inpainting at six random time steps 41
20. HYRAS compared to the daily DDPM model output over the course of the year 42
21. RAPSD of HYRAS vs the daily DDPM model. 42
22. RMSE and Bias of HYRAS vs. Monte Carlo DDPM 45
23. Sparse DDPM inpainting at six random time steps 49

24. Sparse hourly model over the course of the year and day for RMSE, Correlation, Bias and SSIM. 50
25. Spatial correlation (using the Pearson correlation coefficient) of the U-Net against the *RADKLIM* dataset 51
26. RMSE, Bias, Correlation and SSIM over all months of the year for the Monte Carlo Sparse DDPM version 52

List of Tables

1. Patch diffusion parameters. 17
2. Importance Sampling parameters from Ravuri et al. [46] (top) and the two model types used in this project (below). 18
3. Number of patches used per dataset and patch size. 18
4. Hyperparameters for the normal DDPM model. 23
5. Summary of evaluation metrics for the inpainted fields of the hourly model. 33
6. Evaluation metrics for daily inpainted precipitation fields: model output compared to HYRAS. NaN values are ignored. 40
7. Comparison of Inpainted Models and HYRAS across Different Evaluations 44
8. Evaluation metrics for the sparse hourly model compared to the Hourly baseline. 48
9. RMSE of U-Net and Sparse DDPM model variants against the radar data 51
10. Total precipitation difference of U-Net and DDPM model variants compared to the radar data 53
11. RMSE comparison of HYRAS against the ground truth, model output, and improved model output in 2018 53
12. Correlation comparison of HYRAS against model output, improved model output, and ground truth for DDPM and U-Net 54
13. Total precipitation difference between HYRAS and model output, improved model output, and radar 55

List of Algorithms

1. DPS Inpainting 26
2. Training with early stopping via a minimum improvement threshold 29

1. Introduction

On the morning of the 14th of July 2021, the water level of the Ahr river in the south-west of Germany slowly but steadily started to rise. The upcoming rain had been expected, but the immense impact it would have was unforeseen by the local meteorological institutes [49]. The flood would take more than 180 lives and cost more than 40 billion euros [7]

Meanwhile, the rise of artificial intelligence (AI) has led to significant improvements in weather and climate models [61] (see e.g. [19, 27, 9]). However, unlike in other areas such as Natural Language Processing (NLP) and image generation, the available data for climate models is often much more limited.

Hence, elaborate data cleaning and augmentation of available climate and weather data is crucial to create models that can predict catastrophes like the Ahrtal flood earlier and more accurately, and thereby potentially save lives and limit other damage.

1.1. Motivation

The main aim of this thesis project is to analyse how machine learning techniques can be used to estimate missing climate data, in general referred to as inpainting. In particular, it will explore which ML models are promising, what considerations need to be made, how effective they are, and how they compare to traditional interpolation methods.

Specifically, German precipitation datasets will be examined. Complete, high-quality, and high-volume precipitation data are critical not only for elaborate climate models but also risk assessments. At the same time, inpainting of *precipitation* data is a particularly challenging task. It is characterised as non-linear [51], with a right skewed distribution (and is therefore not Gaussian) [59] and highly localised around specific spots [35].

Due to this, machine learning methods might be more promising than traditional statistical approaches. In particular, generative ML techniques, such as image inpainting techniques, have shown significant advancements in recent years, but are underexplored in meteorological applications [27].

After examining the terms inpainting and interpolation in Section 1.2, a brief review of recent work in this field is presented in Section 1.3. Next, the terms, methods and machine learning architectures used in this investigation are explained in Section 1.4. Lastly, in Section 1.5, the scope and layout of this investigation are presented.

1.2. Image Inpainting

The term inpainting originates from the restoration of damaged sections of artwork and was first used in an image processing context in 2000 [3].

It refers to the process of filling areas of an image where data is missing or that are otherwise imperfect. Often, the aim is to make the inpainted area blend smoothly with the surrounding data, making it difficult to distinguish which area was changed.

Interpolation, on the other hand, is a statistical term referring to the process of estimating missing values between known data points, for example in a time series. Hence, both terms describe a similar principle and can be used to refer to the challenge of filling missing values in precipitation data. They will be used interchangeably here.

1.3. Related Work

There are a variety of statistical and machine learning methods that can be used for inpainting/interpolation of meteorological data. An extensive review of *statistical methods* was compiled by Li et al. in 2014 [30]. Some traditional machine learning methods like k-nearest neighbours, random forests and simple neural networks were compared by Militino et al. in 2023 for the purpose of precipitation data interpolation in Spain [37].

In terms of generative methods, Kadow et al. use partial convolutions [27] based on the 2014 paper by Liu et al. [32] to inpaint missing temperature data. There are several approaches for this use case using variants of U-Nets (see [36, 22]) and diffusion models [8].

Further, Bochow et al. [4] use a variant of Suvorov et al.'s inpainting method based on Fourier Convolutions [54] for meteorological data. Graph neural networks (GNNs) are used for this purpose by Huang et al [24].

For spatial downscaling, Mosaffa et al. use U-Nets in their 2025 paper [38] and in the same year Hess et al. showed the efficacy of consistency models for this field [19].

1.4. Preliminaries

Comparing and contrasting all of these methods comprehensively is beyond the scope of this master's thesis. Hence, a focus was set on the *REGNIE* statistical method, a U-Net variant and Denoising Diffusion Probabilistic Models (DDPMs), which are explained in detail in the following section.

1.4.1. REGNIE

Two-dimensional precipitation fields based on station data already exist. The Deutsche Wetterdienst (DWD, German Weather Service) provides the so-called "Hydrometeorologische Rasterdatensätze" (hydro-metereological grid datasets, *HYRAS*) which exist in various spatial (up to 1x1 km^2) and temporal (up to daily) resolutions partly up to 1931 [45].

The method used for the interpolation is called *REGNIE* ('Regionalisierung der Niederschlagshöhen', engl: regionalised precipitation amount) and is explained in

the following paragraphs. It is described in detail in the 2013 paper by Rauthe et al. [45]

Background Field Regression Firstly, the so-called *Background field* $B = \{b_{x,y} | b \in \mathbb{R}^0\}$, where x and y represent cell indices in the field, are calculated. The method uses geographical coordinates and altitude information to improve their regression, assuming that these influence precipitation. Namely, for every grid cell five 'explanatory variables' are introduced:

1. Longitude
2. Latitude
3. Altitude
4. Direction of exposition
5. Slope

The last two parameters can be derived directly from altitude, by considering the altitude around a specific coordinate, hence all variables are derived from coordinates and altitude. The main idea behind the background field calculation uses multiple linear regression in the following formula:

$$y_i = a_0 x_{i0} + a_1 x_{i1} + a_2 x_{i2} + a_3 x_{i3} + a_4 x_{i4} + a_5 x_{i5} + \varepsilon_i \tag{1}$$

Here, i refers to a specific station. x_{i1} through x_{i5} are the values of the five explanatory variables at that station, a_1 through a_5 their coefficients, and y_i the mean precipitation value at that station. The value a_0 is seen as a constant, so x_{i0} is always 1 and ε_i defined as the residual value at the station, the *residuum* in DWD terminology.

All x values are known at all station cells as well as y. If there are n stations considered, then one can consider the matrix $\mathbf{X} \in \mathbb{R}^{n \times 6}$, vectors $\mathbf{y}, \varepsilon \in \mathbb{R}^n$ and explanatory variable vector $\mathbf{a} \in \mathbb{R}^6$ and the equation can be rewritten as:

$$\mathbf{y} = \mathbf{X}\mathbf{a} + \varepsilon \tag{2}$$

The Multiple Linear Regression (MLR) to estimate $\mathbf{a}$ can then be conducted via the ordinary least squares method (provided $\mathbf{X}$ has enough stations n and the matrix can be inverted):

$$\hat{\mathbf{a}} = (\mathbf{X^T X})^{-1} \mathbf{X^T y} \tag{3}$$

Interpolation Method The main idea behind REGNIE is that this regression is calculated on a monthly basis, and for different regions within Germany. These regions are established via an agglomerative clustering of stations with similar values.

The regression for a specific month (e.g. January) and specific a cluster (e.g. a part of Brandenburg) is calculated using data over 30 years. All values for that month of all stations in that cluster are used for the regression. Every month of the year has different clusters and every cluster for that month its own optimised **a** vector.

Subsequently, the background field B is calculated. Let $c = (\hat{x}, \hat{y})$, $c' = (\hat{x}', \hat{y}')$ be cells in the field. $d_{c,c'}$ is defined as the Euclidian distance between both cells. To estimate precipitation at cell c, all we need is the residuum of c, to calculate precipitation using Equation 1.

The residuum of any station i where we have the data can be calculated using y_i. For all other cells, the residuums of the m stations $\varepsilon_1 ... \varepsilon_m$ in a radius of 20 kilometres (30 km if $m = 0$) around that cell are used via inverse distance weighting:

$$\varepsilon_c = \frac{\sum_{1=k}^{m} \frac{\varepsilon_k}{d_{k,c}^2}}{\sum_{1=k}^{m} \frac{1}{d_{k,c}^2}} \tag{4}$$

The daily precipitation fields are then calculated similarly: for every daily station data point, the value is divided by its corresponding monthly background field value. Subsequently, for all other cells a similar quotient is calculated via inverse distance weighting (see Equation 4) and then these quotients are multiplied by the corresponding background fields to yield the final interpolated value.

This method relies on the intuituve assumption that precipitation is heavily influenced by terrain and location. It also exploits the fact that local training for specific regions will be more accurate than general training - for example, in flat areas a storm moves differently than in the mountains.

1.4.2. U-Net

Building on the Deep Neural Network (DNN) autoencoder architecture, Ronneberger et al. developed the so-called U-Net in 2015 [48]. The principle remains the same: data is put into the DNN, an abstraction is produced in the bottleneck, and out of that bottleneck the output is generated again, often in the same dimensions as the input data.

However, the U-Net architecture exhibits several key unique aspects. The general architecture, U-Net Inpainting, and a specific use-case for climate data inpainting with a U-Net Model are discussed below.

Architecture The encoder follows several steps that alternate between increasing channels and decreasing image size. Next, the decoder reconstructs the original dimension (or at least similar ones) by alternating between decreasing channels and

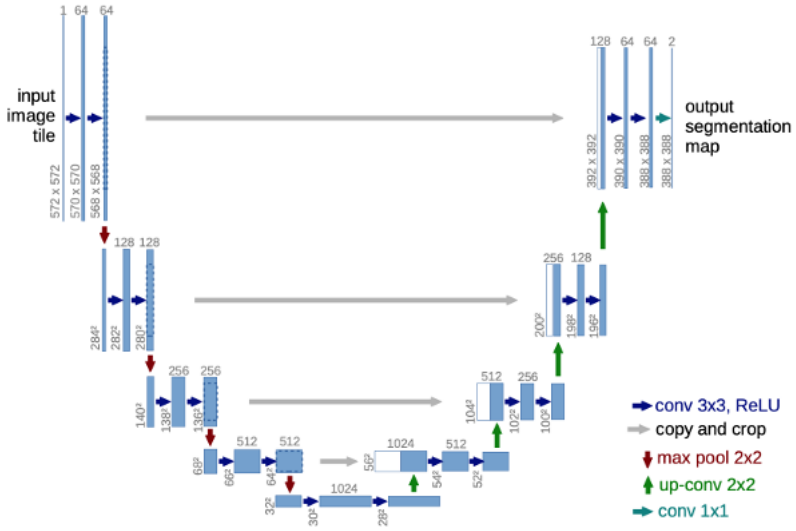


Figure 1: U-Net architecture from the original Ronneberger et al. paper [48]. It consists of the encoder (left), bottleneck (middle bottom) and decoder (right). During encoding, image size is decreased and channel size increased, whilst in the decoder the, the opposite happens, resulting in the typical 'U-shape'.

increasing size. This results in the typical 'U-shape' of the network architecture that can be seen in Fig. 1.

Additionally, so-called skip connections are introduced between the layers of the same dimensions. The last layer of an encoder dimension is concatenated to the first layer of said dimension in the decoder. The authors show that this setup proves very effective for the task of image segmentation and has since been used for various other applications, including image inpainting (see [36, 38, 20]).

Inpainting U-Nets can be used for a lot of tasks such as image segmentation and colourising black and white images in addition to image inpainting. This section examines how the architecture can be used to fill missing data.

The main idea in U-Net inpainting is that during training, the ground truth is masked to obtain data. Ideally these masks resemble the locations of missing data in the datasets for which the inpainting is trained for. This masked data is entered into the network and the loss is calculated in comparison with the ground truth, i.e. the unmasked input.

Traditional approach For image inpainting tasks, the data contains missing values. If they enter the U-Net unmodified, they propagate through the convolutional network architecture and the output becomes unusable.

Hence, authors often fill the missing data with some non-null value, for example the data mean or zero, and add a mask as an extra input channel to the U-Net that tells the model where the valid pixels are (see e.g. [58]).

The intuition is that the model learns that the mask indicates valid values, and then learns the inpainting by masking training data and calculating the loss between the model output and the unmasked image.

Whilst this fixes the missing value problem, filling masks with one value has shown to introduce blurriness and other artifacts to the inpainted image [32].

Partial Convolutions To deal with this issue, Liu et al. introduced *partial convolution* layers [32]. The goal was to avoid introducing any fill values and instead find a mechanism for the net to ignore missing values without these propagating through the network.

Let $\mathbf{W}$ be the weights and $\mathbf{b}$ be the biases of a convolutional layer. Let $\mathbf{M}$ be the mask of valid pixels of the input $\mathbf{X}$ into the layer. The traditional approach implements the mask as an extra channel of the input, here it is a separate matrix. Let $\sum$ be the sum of all values in a matrix, $\mathbf{1}$ be a matrix full of ones and $\odot$ be the dot product. The normal convolutional operation is now $x' = \sum \mathbf{W}^T \mathbf{X} + b$. The partial convolutional operation is instead:

$$x' = \begin{cases} \sum \mathbf{W}^\top (\mathbf{X} \odot \mathbf{M}) \frac{\sum \mathbf{1}}{\sum \mathbf{M}} + b, & \text{if } \sum \mathbf{M} > 0, \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{5}$$

The mask for the next layer is calculated as follows:

$$m' = \begin{cases} 1, & \text{if } \sum \mathbf{M} > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Hence, the main idea is to prevent the missing values influencing the calculation so that within a sliding window, if valid cells are present, their information is scaled up so that the output has a similar magnitude to the rest of the matrix. If enough convolutions happen, so if the U-Net is big enough, in the final convolution m' will be 1 for all pixels. If not, this indicates that the model was too simple or the input was too sparse to generate meaningful outputs, which is in itself a safety check.

The Filippou U-Net Whilst Liu et al. used the approach to inpaint missing information in photos, Kadow et al. applied the partial convolutions first to estimate missing climate data in their much cited 2020 paper [27]. Building on this research, an investigation was conducted by Danai Filippou, supervised by Prof. Kadow on infilling precipitation data with U-Nets. The research has been presented at a conference [13] and resulted in a master's thesis that can be found via Appendix B. It is discussed in this section.

The structure of the U-Net used in their work is shown in Fig. 2. The main differences with the traditional U-Net are that it uses partial convolutions and that skip connections are also introduced between the max pooling and upsampling steps. The channel number is doubled in every layer (except for the initial channel number which is 137), and height and weight are halved, leaving a shape of $(C \times H \times W) = (1100, 18, 15)$ in the bottleneck.

Another important difference is the loss function used for the actual back propagation in training. The original U-Net was used for image segmentation problems in biomedicine [48] which is essentially a classification problem. Hence, the authors use cross entropy loss to optimise the image segmentation network.

Here, contrarily, the task is to recreate the ground truth precipitation field so the natural choice for a loss function would be to look at pixel vs. pixel comparisons, like in the mean average error (MAE). MAE (also called L^1), defined as $L^1 = \frac{1}{n}\sum_{c \in C} |O_c - G_c|$, where C is the set of cells in both the output O and the ground truth G.

Filippou uses the same method as Liu et al. here and calculates loss a mixture of L^1 (differentiated by MAE in the valid area L_{valid} and the missing area L_{hole}, with perceptual loss $L_{perceptual}$ as defined by Gatys et al. [15]), a newly defined style loss L_{style} and the total variational loss [26]. For detailed definitions, the referenced literature can be consulted.

The loss terms aim to achieve pixel to pixel accuracy and produce similarly structured and varied data. The final loss is calculated using coefficients, these hyperparameters have been optimised based on subjective interpretation of 100 validation images.

$$L_{\text{total}} = L_{\text{valid}} + 6\, L_{\text{hole}} + 0.05\, L_{\text{perceptual}} + 120\, L_{style} + 0.1\, L_{\text{tv}}. \tag{7}$$

1.4.3. Denoising Diffusion Probabilistic Models

One of the most promising developments in image inpainting in recent years are Denoising Diffusion Probabilistic Models (DDPMs) [34]. They were developed in 2020 by Google for image generation tasks [20]. The basic principle is to start with noise and iteratively create an image from there.

Idea During training, noise is added in a fixed number of steps to an image until Gaussian noise is created, the so-called *forward process*. Then a neural network, usually a U-Net, is trained to predict the noise that was added to an image at a specific step.

The *backward process* starts with Gaussian noise. The trained network predicts the noise that was 'added' in the hypothetical previous step and subtracts that from the given noise. Iteratively, the model learns to recreate each training image from noise. The process is portrayed in Fig. 3. Therefore, when starting with random noise, a completely new image can be generated via this process. Abstractly speaking, the

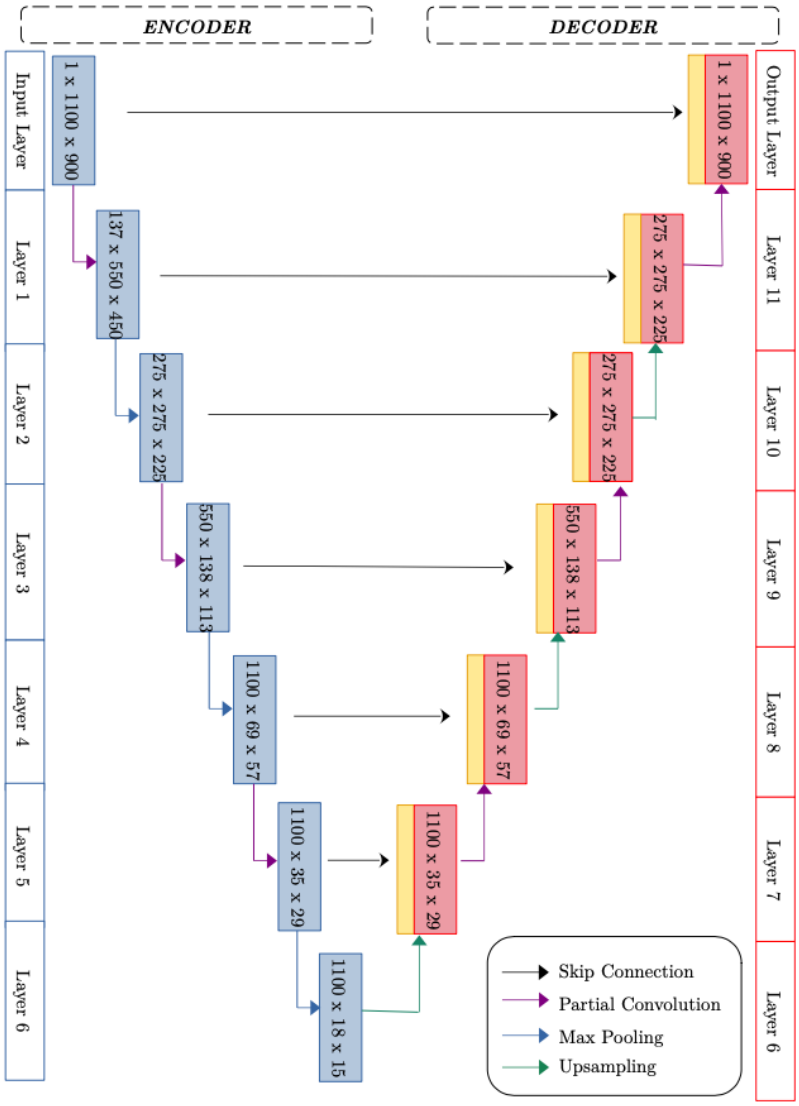


Figure 2: U-Net architecture used by Filippou et al. [13]. The yellow boxes represent the feature maps passed from encoder to decoder via the skip connections. The use of the term layer here is slightly different from the normal U-Net context, where a convolutional operation connected with pooling or upsampling is implemented as a single layer with one skip connection, not two. An equivalent normal U-Net would be considered to have 6 layers, as there are 6 convolutions, albeit only half of the skip connections.

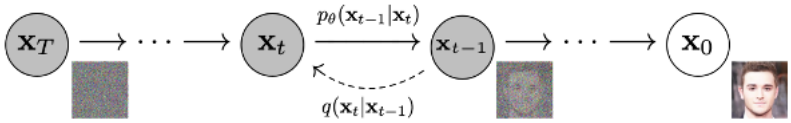


Figure 3: DDPM forward and backward diffusion process (from [20]).

model learns the distribution of plausible rain fields out of the possible values the pixels in an image can take.

For image inpainting, the masked area is filled with random noise, and the network subtracts the noise in each step, whilst simultaneously recreating the original picture in the unmasked area.

Realization Formally speaking, during the actual diffusion (adding noise to the image), also called the *forward process*, a Markov chain over T time steps (typically $T = 1000$) from an image x_0 to pure noise x_T is realised. As it is a Markov chain, each x_t only depends on x_{t-1} and so the conditional distribution $q(x_t \mid x_{t-1})$ can be defined. This is done by Ho et al. using a so-called beta-schedule, a pre-defined time series of variances via Eq. 8 [20]:

$$q(x_t \mid x_{t-1}) = \mathcal{N}\Big(x_t\,;\ \sqrt{1-\beta_t}\,x_{t-1},\ \beta_t\mathbf{I}\Big) \tag{8}$$

Therefore, in a single step q_t is sampled from a distribution with mean $\sqrt{1-\beta_t}\,x_{t-1}$ and variance $\beta_t\,\mathbf{I}$ where $\mathbf{I}$ is the identity matrix. The beta variance schedule b_t is chosen to linearly increase from 0.0001 to 0.02 over the T time steps. Consequently, the variance increases and the signal from the original image decreases over the course of the diffusion process.

Contrastingly, in the *reverse process* noise is removed from a completely noisy image. It is again implemented as a Markov chain $p(x_{t-1} \mid x_t)$. The process uses a model θ, usually a U-Net, to estimate the noise ε that was added in time step $t-1$ to obtain $q(x_t \mid x_{t-1})$. The resulting estimate is defined as $\varepsilon_\theta(x_t)$. The reverse process is again a distribution for each time step as random noise was added in the forward step. It is defined by the authors using Eq. 9.

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\Big(x_{t-1}; \mu_\theta(x_t,t), \Sigma_\theta(x_t,t)\Big) \tag{9}$$

So a x_{t-1} is sampled from a distribution with a mean $\mu_\theta(x_t,t)$ that depends on the image at time step t and the time step itself. The same is the case for the variance $\Sigma_\theta(x_t,t)$ of the distribution.

Before continuing, the helper variables α_t and $\hat{\alpha}_t$ are defined for simplification in Eqs. 10 and 11.

$$\alpha_t = 1 - \beta_t, \tag{10}$$

$$\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s \tag{11}$$

The mean is estimated using the model θ via its error estimate $\varepsilon_\theta(x_t)$ using Eq. 12.

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \varepsilon_\theta(x_t, t) \right) \tag{12}$$

Here the authors apply a trick and simply assume the variance to be equal to the beta schedule as shown in Eq. 13.

$$\Sigma_\theta(x_t, t) = \beta_t \, I \tag{13}$$

Lastly, x_{t-1} can finally be sampled from x_t using Eq. 14.

$$x_{t-1} = \mu_\theta(x_t, t) + \sqrt{\beta_t \, I} \, z, \quad z \sim \mathcal{N}(0, I) \tag{14}$$

The derivation of the formulas is described in-depth in the initial paper by Ho et al. [20], various aspects relating to the implementation in this investigation are discussed further below.

Use DDPMs have already been applied successfully to precipitation datasets for the purpose of nowcasting [40], essentially 'predicting' the *current* rainfall in a given area, as data is not usually immediately available.

The advantage of DDPMs over other inpainting techniques in a context like precipitation is that they are *generative* and also *non-deterministic*, which can be an advantage. Non-generative techniques might be too imprecise and linear for precipitation data as it has such high local extremes.

However, generative deep learning (DL) techniques do not learn semantic concepts directly and therefore might produce unrealistic or seemingly random results (similar to AI-hallucination). As DDPMs are dependent on some random noise in the beginning of the inpainting process, changing that noise should still produce equally realistic but different results for any Gaussian noise. Hence, they are non-deterministic.

The problem of results that appear unrealistic as they defy meteorological laws could be counteracted by producing various results and getting a probability distribution for precipitation in a specific location. The underlying hypothesis is that, while individual samples may be unrealistic, aggregating multiple generated predictions can mitigate such errors and yield a statistically robust estimate of the precipitation distribution at a given location. This is an advantage that DDPMs have over e.g. U-Nets.

1.5. Aim

The main purpose of this investigation is to assess how well DDPMs can be used for climate data inpainting, test whether they are better suited for non-Gaussian data such as precipitation data, and compare them to other machine learning and statistical methods.

To this end, three experiments were conducted:

1. A DDPM model based on hourly precipitation data was trained and compared to the ground truth.
2. A DDPM model was trained with daily precipitation data and compared to the HYRAS dataset based on the REGNIE method.
3. A sparse hourly model, using only data from a subset of available stations was developed and compared to a U-Net based inpainting method as this situation replicates the historic station coverage.

The data methods used in DDPM training and inpainting as well as the evaluation tools used are explained in Section 2. This is followed by the chapters on the three experiments, each including the results and a discussion in Sections 3 - 5. Lastly, the experiments are discussed in the general context of the field in Section 6 followed by a conclusion in Section 7. In Appendix A the code is linked via a GitHub repository.

2. Methods

The general idea of how DDPMs work was discussed in Section 1.4, but there are various design choices for how exactly the training, sampling and inpainting are implemented. Furthermore, there are various aspects of the data used that can significantly influence the results and should therefore be noted. This chapter covers the data used (Section 2.1) as well as the model design (Section 2.2).

2.1. Data

German precipitation data collection goes back hundreds of years and is primarily coordinated by the German weather service, DWD. The following Section 2.1.1 covers both the primary data sources as well as the datasets used for inpainting comparison.

Next, the data preparation is presented in Section 2.1.2. There, various aspects of data cleaning, scaling and segmentation that are useful and partly necessary for DDPMs specifically are discussed.

2.1.1. Sources

There are two primary precipitation data sources and one inpainting comparison dataset used that are examined in depth in the following paragraphs.

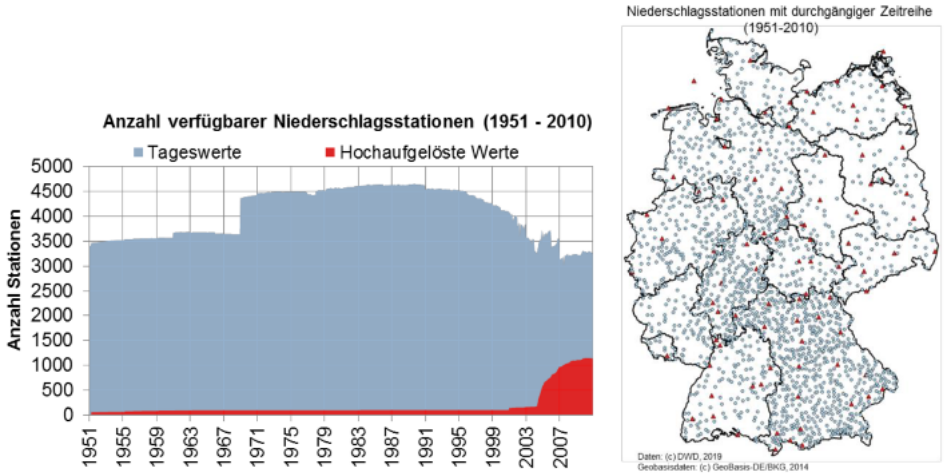


Figure 4: Magnitude and density of precipitation stations in Germany from [39]. Left: Amount of available precipitation stations with daily (blue) and sub-daily (red), so for example Mannheim hours, hourly, sub-hourly, resolution. Right: Distribution of such stations with <10% missing values across Germany (blue circles: daily, red triangles sub-daily).

Station Data The ground station network of the DWD covers around 400 active stations [11]. Additionally, several thousand stations from cooperating partners are encompassed in the precipitation station datasets. These are published in the DWD climate data centre (CDC) database [12].

In the ground stations, precipitation is usually estimated in rain gauges (also known as *ombrometers*) that collect rainwater with a funnel and measure this input with a weighting scale [13]. Historically, these were recorded manually, in daily or sub-daily intervals often in the so-called *Mannheimer Stunden* (Mannheim hours) at 7:00, 14:00 and 21:00 [2].

With regard to data quality, manual collection, often done by volunteers entails some degree of uncertainty, and there are some gaps in the data. Whilst electronic precipitation sensors were rolled out in the late 1980's these also sometimes malfunction [13] so these aspects should be considered when using the data for modelling.

There are around 3,500 stations across Germany which provide a daily precipitation time series at least up to 1951 [39]. Hourly time series, on the other hand, are only available for 76 stations [13] reaching back that far, whilst they more widely spread out in the mid-2000s to about 1000 stations [39], see Fig. 4.

Assuming precipitation is roughly similar in the 1 km^2 range around a ground station, with Germany's 358 thousand kilometres squared, this yields a station coverage of only about 1 % for daily stations, 0.3% for hourly stations in the last twenty years and just over 0.02% for twentieth century hourly stations.

Image inpainting techniques were often developed for datasets with much higher coverage [34, 63]. This highlights the difficulty of the inpainting task.

RADKLIM The *RADKLIM* dataset [60] by the DWD provides precipitation data measured by radar from 2001 on. Measurements are given on a 1x1 km^2 grid in a hourly temporal resolution in the NetCDF format. It covers a 1100x1200 km^2 (height x width) grid including Germany and neighbouring countries. The area covered is shown in Fig. 5 where the location and range of the 16 ground radars of the network are also shown.

The dataset is highly sparse, with 64.1 % missing values, not only due to the range covered substantially exceeding Germany's borders but also because the radar stations often misperform or the signal is obstructed. Radar measurements are indirect, inferring the precipitation rates from the pulses sent back from rain, snow or hail after emitting an electromagnetic signal.

There are a variety of factors impacting radar data availability and quality. These include interference through mountains, buildings, windmills or birds, as well as radome wetness and cleanliness. An overview of factors decreasing radar data quality is provided in Fig. 6 ([23, 13]).

The DWD counters these issues through a series of data cleaning methods including noise and clutter reduction as well as adjusting the reflectivity rate to ground station measurements. The service offers various precipitation products based on these techniques. A detailed description is provided in [60].

For this investigation, the DWD dataset *RADKLIM* (RADar KLIMatologie) is chosen due to no interpolation methods being used to fill missing data in it which might falsify the model. Further, it has been calibrated with rain gauge data, improving data quality.

Whilst several drawbacks of radar precipitation data have been outlined, the *RADKLIM* dataset is viewed as the ground truth for the purposes of this investigation as no better high resolution (neither spatial nor temporal) precipitation field data is available.

HYRAS For the second experiment, the DDPM model is compared to the REGNIE method described in Section 1.4.1. The latter resulted in the *HYRAS* dataset, that is described by Rauthe et al. and updated iteratively by the DWD [45]. The dataset covers a grid of 890 x 665 km^2 over Germany on a 1 km^2 resolution and is available from 1931 on on a daily resolution.

The dataset is used as a comparison in Section 4 for the model output in the year 2018.

2.1.2. Data Preparation

The *RADKLIM* training data was split into training (2001-2011), validation (2012-2017) and test (2018) data (in line with Run 1 of the U-Net comparison model [13]).

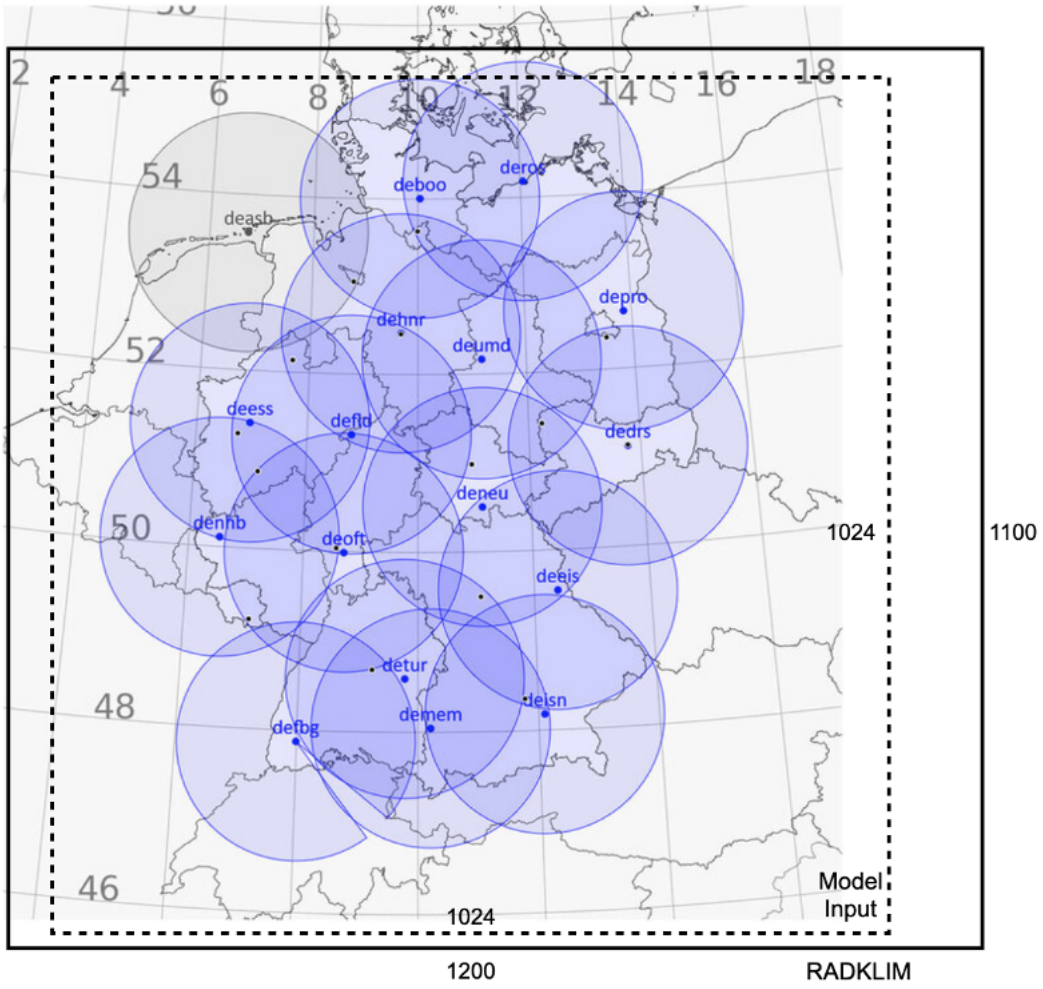


Figure 5: Location and range of the DWD radars [53] amended with the spatial scope of the RADKLIM dataset and the 1024x1024 km2 projection used as a model input for the DDPM.

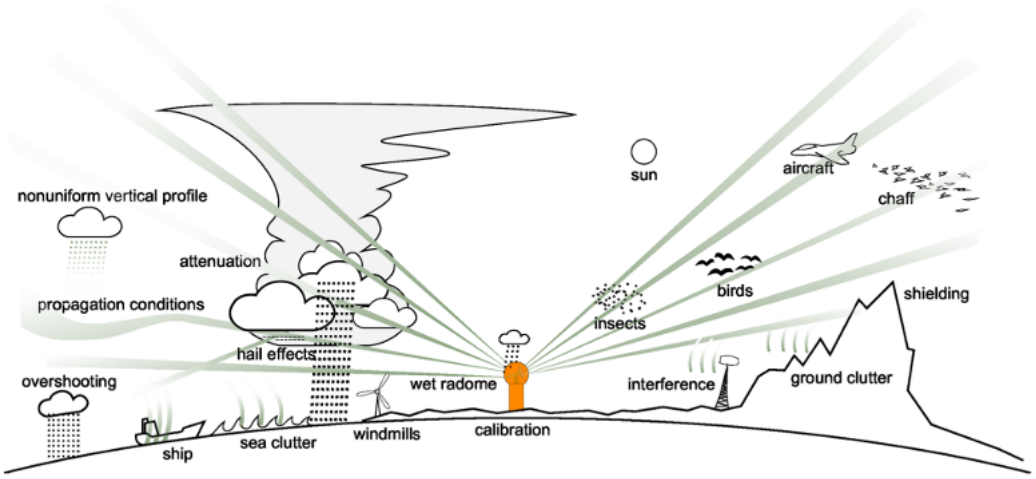


Figure 6: Various factors can decrease radar quality (from [23], see [13])

The 1100 x 1200 km^2 RADKLIM grid is limited to a 1024 x 1024 km^2 over Germany (see Fig. 5) as $1024 = 2^{10}$, so the number is easily divisible by two which is both helpful in U-Net architectures in general (because during pooling in the encoder channel number is doubled whilst height and width are halved, and the opposite for the up-convolutions in the decode, see [48]). It is also necessary for *Patch Diffusion* (see next paragraph). Lastly, the RADKLIM arrays are empty and outside the scope of the radars in Fig. 5 so no information is lost.

For the daily model, the data is aggregated. Any negative values, which could be due to technical errors, are zeroed. For the output, the data was remapped to the HYRAS grid to make all formats comparable. Here, every cell in the HYRAS grid was assigned the value of the closest cell in the Radar / U-NET grid.

This is due to station coordinates and the coordinates of the centres of the cells points in the different grids not overlaying exactly. The problem is illustrated in Fig. 7. Cells outside of Germany are ignored as the investigation focuses exclusively on Germany.

Patch Diffusion As Wang et al. point out, "Diffusion models are powerful, but they require a lot of time and data to train." [56]. Whilst diffusion models were initially introduced to generate relatively small sized images, they can also be used for huge fields and various channels.

This can quickly significantly increase run time and storage space which in turn causes high energy consumption and impact on the environment [25]. When input field height and width are doubled, memory quadruples. Iyengar et al. show that runtime (measured in FLoating point OPerations [FLOPS]) in convolutions, that form the basis of U-Nets, is proportional to height and width as well as shown in Eqs. 15.

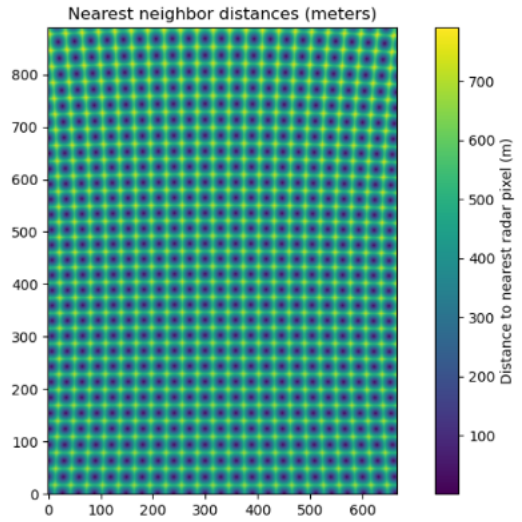


Figure 7: Distances between HYRAS grid coordinates and the assigned *RADKLIM* grid coordinates, estimated with the haversine function that is slightly inaccurate as it assumes the Earth to be a perfectly round sphere.

$$\begin{aligned} \text{FLOPs}_{\text{conv}}(k, C_{in}, C_{out}, H, W) &= 2 \cdot H \cdot W \cdot k^2 \cdot C_{in} \cdot C_{out} \\ \therefore \text{FLOPs}_{\text{conv}} &\propto H \cdot W \\ \therefore \text{FLOPs}_{\text{conv}}(k, C_{in}, C_{out}, 2H, 2W) &= 4 \cdot \text{FLOPs}_{\text{conv}}(k, C_{in}, C_{out}, H, W) \end{aligned} \tag{15}$$

So whilst technically training sixteen 256 x 256 km ^{2} patches requires the same total time and total energy, GPU memory can break with bigger patches, and there are further non-linear effects that make training bigger patch sizes more time consuming, outlined in [56] (e.g. the use of self-attention layers).

Therefore, Wang et al. propose *Patch Diffusion*, a technique where a diffusion model is trained with training data patches of varying sizes. This works as convolutional neural network architectures have *translation equivariance* - they use the same filter everywhere, so they work on input sizes 64x64 km ^{2} and 1024 x1024 km ^{2} in the same way, for example.

This has the further benefit that when looping through the patches during training for small patches, a high degree of local detail can be learned by the model, whereas for bigger patches, correlations across greater distances can be considered by the model. If the same weight is assigned to each patch during training, this disproportionately favours smaller patches which in this case is intentional.

In this thesis study, an adaption of the *Patch Diffusion* technique is implemented where the input data time steps were split randomly according to data shares to three patch sizes, the 1024x1024 fields were split into quadratic sub-patches of the given patch size using a constant stride.

Patch Size	**64**	**128**	**256**
Data Shares	10 %	30 %	60 %
Stride	64	64	64

Table 1: Patch diffusion parameters.

Patches containing missing values were ignored which results in the loss of some valuable information. Alternatively, the values could be set to 0 and another mask channel could have been added as described in [32]. However, this would have made the network slower. Using the same small stride even for bigger patch sizes ensures that a single missing value does not cause disproportionate data loss (even though this does lead to some duplicates). The parameters used for patch diffusion are outlined in Table 1.

Importance Sampling Precipitation data distribution is, as described, very right skewed, having a peak at zero and with medium low values and very little high values. This leads to a lot of patches being created that consist mostly of zeroes. Whilst it is good that the model learns this distribution, those patches contain very little structural information about precipitation dynamics. Therefore, it would be beneficial if patches with precipitation could be prioritised during training.

The same issue was described by Ravuri et al. when using Generative Advesarial Networks for precipitation nowcasting (estimating the current precipitation based on very recent data, potentially forecasting a few hours into the future) in Britain [46].

The authors solve this issue by developing the so-called *Importance Sampling* scheme that has been adapted for this project as follows. After creating the patches for the different patch sizes and filtering out the ones with missing values, a saturated rain value $x_{n,c}^{sat}$ in a cell c of a patch n is computed from rain rates x using a saturation constant s with this sigmoid-like formula:

$$x_{n,c}^{sat} = 1 - e^{-x_{n,c}/s} \tag{16}$$

Subsequently, an *acceptance probability* q_n for a patch n is calculated. Let C_n be a set of cells in n, $q_{min} \geq 0$ a minimum probability and m a factor setting the overall inclusion rate.

$$q_n = min(1, q_{min} + \frac{m}{|C_n|} \sum_{c \in C_n} x_{n,c}^{sat}) \tag{17}$$

The resulting acceptance probability for a patch n is then used to determine if the dataset is used. The authors set the values of the parameters s, m and q_{min} differently depending on each dataset and use (training, validation, testing) and further define a method to produce randomly sized crops which is not employed in this project.

Dataset / Model	s	m	$q_{\min}$
Training UK	1.0	0.1	$2 \cdot 10^{-4}$
Training US	1.0	0.1	$2 \cdot 10^{-4}$
Validation UK	1.0	2.2	$5 \cdot 10^{-3}$
Validation US	1.0	0.2	$5 \cdot 10^{-3}$
DDPM Hourly	1.0	0.5	$5 \cdot 10^{-3}$
DDPM Daily	1.0	2.2	$5 \cdot 10^{-3}$

Table 2: Importance Sampling parameters from Ravuri et al. [46] (top) and the two model types used in this project (below).

To make this model easily comparable to other models, importance sampling is not used for test data, and the same parameters are used in training and validation data. The parameters used by the importance sampling authors are portrayed in Table 2 alongside the parameters used in this investigation. They were optimised manually to find a good balance between dataset size and runtime length.

These chosen values result in about 50 % of the daily dataset being used whereas only 2-3 % of the hourly dataset is used. As there are 24 times as many hourly datasets available, this results in a comparable magnitude of training and validation data. As the hourly dataset is more selective, the resulting training dataset also contains less empty data. The exact amounts are shown in Table 3.

Dataset	64×64	128×128	256×256
Hourly Training	19 630	41 566	31 372
Hourly Validation	10 855	21 457	14 802
Daily Training	13 451	29 371	20 536
Daily Validation	8 427	16 823	10 882

Table 3: Number of patches used per dataset and patch size.

This approach consequently leads to the described bias of training and validation data towards higher precipitation. The authors counteract this by using the inverse acceptance probability in the loss function. Given that the original dataset could have consisted of patches $x_1, x_2, ..., x_n$ they estimate its loss $\sum_{i=0}^{n} S(x_i)$ with the subset $x_{a1}, x_{a2}, ...x_{ak}$ selected via importance sampling with the equation:

$$\sum_{i=0}^{n} S(x_i) \approx \sum_{j=0}^{k} S(x_{aj} q_{aj}^{-1}) \tag{18}$$

Hence, samples that had a lower probability of entering the used dataset are boosted in loss calculation and vice versa. The authors did not employ this during gradient calculation in training as they "found no significant advantage"[46] to this so the loss correction is only applied in validation in this project.

Scaler DDPMs were specifically chosen for this project because of their potential to yield better results in the difficult feat of interpolating/inpainting non-normally distributed data. However, they still use Gaussian noise in the forward diffusion process and as a basis for the reverse process.

More specifically, in every forward step during training, a little bit of Gaussian noise is added to the data, randomising the input. As described, the model then learns to reverse that process, estimating the 'noise' in the patch and subtracting it to obtain the signal/original. When sampling (or inpainting), the model starts with complete Gaussian noise and consecutively samples a less noisy patch.

Hence, if the data is more Gaussian, it might be even easier for the model to learn the desired distribution. In this project, the data is therefore standardised and then normalised before entering the model.

For normalisation, the standard z-score transformation was used: the training data's mean $\mu = \frac{\sum_{i=1}^{n} x_i}{n}$ and standard deviation $\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n}}$ were calculated and then every scaled data point $\hat{x}$ transformed using Eq. 19. This way the data has a mean of zero and standard deviation of one which is necessary for DDPMs.

$$z = \frac{x - \mu}{\sigma} \tag{19}$$

For standardisation, three functions were considered:

- *Log transformation* where $y = ln(\frac{x}{c} + 1)$
- *Asinh transformation* where $y = arcsin(\frac{x}{c}) = ln(\frac{x}{c} + \sqrt{(\frac{x}{c})^2 + 1})$
- *Box-Cox transformation* where $y = \frac{x^\lambda - 1}{\lambda}$, or $y = ln(x)$ if $\lambda = 0$, see [6]

Here, the parameters c and λ are optimised to statistically make the data resemble a Gaussian distribution as much as possible. All the transformations were applied to a subset of the data and subsequently normalised. The results can be seen in Fig. 8.

The *Box-Cox transformation* is evidently the least normal, whilst the other two perform similarly. The *log transformation* was chosen due to its simpler formula, with $c = 0.0625$ as the value that had a skew and kurtosis most similar to a Gaussian distribution.

2.2. Model Design

In this section, first the architectural considerations of the U-Net model are described in Section 2.2.1, building on which the hyperparameter optimisation is described in Section 2.2.2 and lastly, in Section 2.2.3 the sampling and inpainting implementation is discussed.

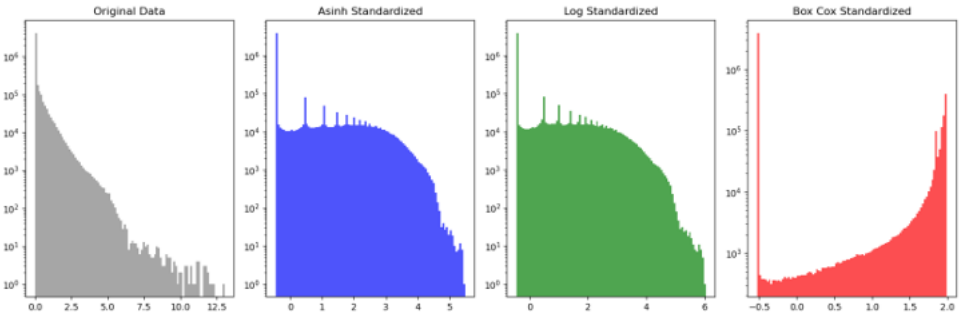


Figure 8: Radar precipitation data (left) transformed with *Asinh*, Log & *Box-Cox* standardizers. The results are on a log scale due to the huge peak at 0 in the original distribution.

2.2.1. Architecture

Denoising diffusion models have been introduced in 2020 [20]. Since then, various DDPMs have been implemented and a range of libraries and existing models are available, like NVIDIAs EDM Model [28] or Inpainting specific ones (see [63, 34]).

As this investigation aims to explore the efficacy of DDPMs as a model choice for climate inpainting-related tasks in general, the goal was to create a model that sticks to the default DDPM design as described in [20] rather than having a hyper-optimized variant. Hence, the model was designed in Python, mainly using the PyTorch library [44], following the formulas in the paper.

The U-Net itself which is the basis of the DDPM model has not been reimplemented from scratch. Here, the version used in NVIDIAs EDM model [28] which was a reimplementation of the U-Net used by Song et al. [52] which in turn is an improved version of the original DDPM paper [20] and called *DDPM++*. It is usable for both DDPM and score-based diffusion (s. [52]).

The traditional U-Net consists of several layers. During the downsampling stage, the input is downscaled using 2x2 max-pooling (or similar) whilst the number of channels is often increased [48]. For denoising models, there are a series of architectural options that can be integrated into the U-Net to improve learning the distribution.

Song et al.'s DDPM++ architecture amends the Ronneberger U-Net with several features, the most important of which are explained in the following paragraphs.

The U-Net blocks are transformed into *Residual blocks* so that the input to a block is added to the output of the block. Abstractly, $y = f(x) + x$, so instead of learning the desired output y directly, the model is learning $y - x$, the noise in the output. This is based on the *ResNet* paper, where He et al. introduce REsidual (neural) NETworks and show that they are easier to optimize and more accurate with increasing depth [18]. Song et al. argue that this is beneficial in denoising models [52].

At lower layers and hence lower resolutions, *attention layers* are introduced at the

skip connections of the U-Net that aim to reduce redundant feature extractions [55]. After first experiments, it showed that they increase runtime substantially so they were not used in this project.

The authors also use the Sigmoid Linear Unit *SiLU activation function* rather than ReLUs. SiLU is basically sigmoid of x multiplied with x so $SiLU(x) = x \cdot \frac{1}{1+e^{-x}}$. The function looks very similar to ReLUs but has a smoother gradient rather than a jump at $x = 0$.

Further, an *embedding of the time step* t is concatenated to every layer (here the Sinusoidal Positional Embedding [28]), and values are normalized using *Group Normalization*, where channels are organized into groups and then the values in one group is normalized rather than in one batch. This helps when using small batch sizes [62] as done in this project due to memory limitations in such a big network.

Lastly, *residual encoders* are introduced as an option where a parallel encoding stream is implemented in which the original input is subjected to a single downsampling convolution at each layer. This auxiliary patch is then added to the layer output (and multiplied with $\frac{1}{\sqrt{2}}$ to keep the scaling roughly similar). This is due to the characteristic of deep U-Nets to transform the original sample so intensely that in the beginning of training, when weights and biases are randomised, the signal can get so reduced that it gets hard to recover in the decoder.

2.2.2. Hyperparameter Optimization

Whilst the general DDPM and U-Net setup is given, there are various architectural decisions to make. Learning success can highly vary depending on the chosen configurations like which loss function is used or how many layers the model has. In this section, these options are discussed in relation to the precipitation inpainting challenge, also considering constraints like runtime and memory usage.

Parameters The parameters used can be subdivided into the broader categories *U-Net*, *DDPM* and *Training* that are discussed in the following paragraphs.

U-Net The '*Song U-Net*' used has a range of parametrization options for DDPMs out of which most are listed here. For the others, the default values provided were regarded as sufficient.

- *Model channels* - the base number of channels introduced in the first layer
- *Layers* - the number of layers
- *Channel multiplications* - the factors by which the base channel number is multiplied in each layer going to the next 'lower' one
- *Block number* - Number of residual blocks in each layer

- *Dropout rate* - the rate of activations chosen at random that are zeroed at every forward pass during training, to stabilize it
- *Downsample type* - whether to use residual encoder layers or not
- Attention layers - at which resolutions attention layers should be implemented

As mentioned, attention layers are not used in this project. Similarly, whilst technically the model can extract more features, the more model channels are included, using too high number of initial model channels caused the model to run over a week, 128 was chosen as a middle ground.

DDPM In terms of DDPM architecture, the two main aspects are number of *time steps* and the chosen *beta schedule*.

In terms of the former, $T = 1000$ is often chosen as the default option in literature (e.g. [20]). Whilst different values were tested, 1000 showed to be a good balance between complexity and efficiency.

For the latter, it is first necessary to determine the intended β values at the first and last time step. In this project, the standard $\beta_1 = 0.0001$ and $\beta_T = 0.02$ were chosen (see [20]).In terms of the change of beta over the time steps, three alternative schedules were considered.

- Linear: $\beta_t = \beta_1 + (\beta_T - \beta_1)\frac{t-1}{T-1}$
- Quadratic: $\beta_t = \left(\sqrt{\beta_1} + (\sqrt{\beta_T} - \sqrt{\beta_1})\frac{t-1}{T-1}\right)^2$
- Exponential: $\beta_t = \beta_1 \left(\frac{\beta_T}{\beta_1}\right)^{\frac{t-1}{T-1}}$

The schedules differ in how much noise is added at each step, in the linear schedule, it is fairly consistent, the quadratic schedule is slower in the beginning and faster in the end whereas the exponential schedule is even more extreme.

Often, also a cosine beta schedule, introduced by Nichol and Dhariwal in 2021 [43], is used. Here, Eq. 20 is used, which might be an interesting option in future implementations.

$$\beta_t = 1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}, \quad \bar{\alpha}_t = \frac{\cos\left(\frac{t/T+s}{1+s} \cdot \frac{\pi}{2}\right)^2}{\cos\left(\frac{s\pi}{2}\right)^2} \tag{20}$$

Training In terms of training, the parameters to configure here are the standard machine learning aspects loss function, scheduler, optimizer and learning rate.

In DDPM research, mean squared error (MSE), so the mean squared difference between the noise ϵ added at time step t and the noise predicted by the model, so in this case $\frac{1}{n}\sum_{i=1}^{n}(\epsilon - \epsilon_\theta(x_t, t))$.

Considering optimization, the standard *Adam* [29] and *AdamW* [33] optimizers were compared. For learning rate schedulers, an exponential learning rate with $\gamma = 0.99$ and a warmup cosine schedule with 3 warmup steps were considered.

Optuna The optimization was conducted using the Optuna framework. While a traditional grid search is possible with the framework, it is in this case infeasible. Hence, Optuna defaults to a more elaborate *design-by-run* principle where it starts with a random parameter configuration test (in Optuna terminology *trial*) and before every new trial the configuration is set based on the success of past trials [1]. It starts with a random configuration. The ranges of parameters tested and optimization results from Optuna are shown in Table 4.

Parameter Name	**Possible Range**	**Optimized Value**
num_blocks	1 – 3	3
dropout	0.0 – 0.3	0.2704
downsample_type	residual, standard	residual
channel_mult	124, 1224, 1248, 1124	1124
beta_schedule	linear, exponential, quadratic	linear
optimizer	Adam, AdamW	AdamW
scheduler	ExponentialLR, WarmupCosine	ExponentialLR
lr	1×10^{-5} – 5×10^{-4}	1.871×10^{-4}

Table 4: Hyperparameters for the normal DDPM model.

2.2.3. Sampling and Inpainting

Next, it is discussed how the DDPM generates fields of the learned distribution and actually implements the inpainting.

As discussed, sampling in DDPMs is done by creating random Gaussian noise and then creating an image (the sample) in the learned distribution by using the reverse process, continuously subtracting the noise the model estimates over all time steps.

During inpainting, some cells of x_0, the output image, are known. Hence, inpainting can be achieved by overwriting the sample at every time step with the known cells. Two problems can arise here that are discussed in the next paragraphs along with the implemented solutions.

Clamp Range Firstly, during DDPM training the model always only learns to estimate the noise added at a specific time step. The model might learn to nudge a cell of x_t to a higher value so that it becomes part of the learned distribution (because it estimated the noise added to have been negative). If this happens at all 1000 time steps, the value in the cell can fall way outside the range of training values.

This is encountered by introducing a clamp range. After each update in the reverse process, the values in the cells of x_{t-1} are clamped to a range (c_t^{min}, c_t^{max}) so that it is guaranteed that the values do not fall outside this range.

The range changes for every t, because if only one range was applied it would either be to wide (larger than the training data) or the clamping would cause the values to already be within the desired range during the early reverse process so the model can nudge values only away from the boundaries so the resulting precipitation values are only in a middle range.

The clamp range is calculated in this investigation as follows. Firstly, the clamp range at the start of the reverse process is chosen $(c_{start}^{min}, c_{start}^{max}) = (-3.5, 5)$. These values were selected experimentally, given that the data used in the model is normalized and the training data is within a range of about $(-0.3, 3.3)$.

Next, the clamp range at the end of the reverse process $(c_{end}^{min}, c_{end}^{max})$ is calculated. Let T be the set of values $x_i \in T$ in the training data. The clamp range at the end ($t = 0$) is calculated using Eqs. 21. The lower boundary is just the minimum value in the training data.

$$\begin{aligned} c_{end}^{min} &= \min(T) \\ x_{max} &= \max(T) \\ x_{qh} &= x_{99.9} + 0.5 \cdot \sigma \\ x_{ah} &= \frac{1}{2}(x_{max} + x_{max}) \\ c_{end}^{max} &= \max(x_{qh}, x_{ah}) \end{aligned} \tag{21}$$

The upper boundary uses the 99.9th quantile $x_{99.9} = q(T, 0.999)$ as well as the standard deviation $\sigma = \mathrm{std}(T)$ to calculate a quantile-based high x_{qh} value. If the maximum value in the dataset is higher than the quantile-based high, the average of them is used as this indicates that the maximum value is an outlier. This way a compromise was found between allowing the model to estimate high values, potentially higher than in the training data, and not allowing outliers to dominate the final range of values.

Finally, (c_t^{min}, c_t^{max}) are calculated by linearly interpolating $(c_{start}^{min}, c_{start}^{max})$ $(c_{end}^{min}, c_{end}^{max})$ using t.

To leave the model some degree of freedom, the clamping is skipped for the last five time steps of the reverse process ($0 \leq t < 5$), so the estimated values can cross the clamp range, but they are less likely to exceed the range significantly.

Diffusion Posterior Sampling The second problem arising concerns the inpainting process described. When overwriting the known cells at every step of the reverse process, known as *hard conditioning*, the surrounding cells might still differ a lot in an unrealistic way.

To counter this, Chung et al. proposed a method called *Diffusion Posterior Sampling* [10]. The idea is best explained visually. Figure 9 is a representation of the

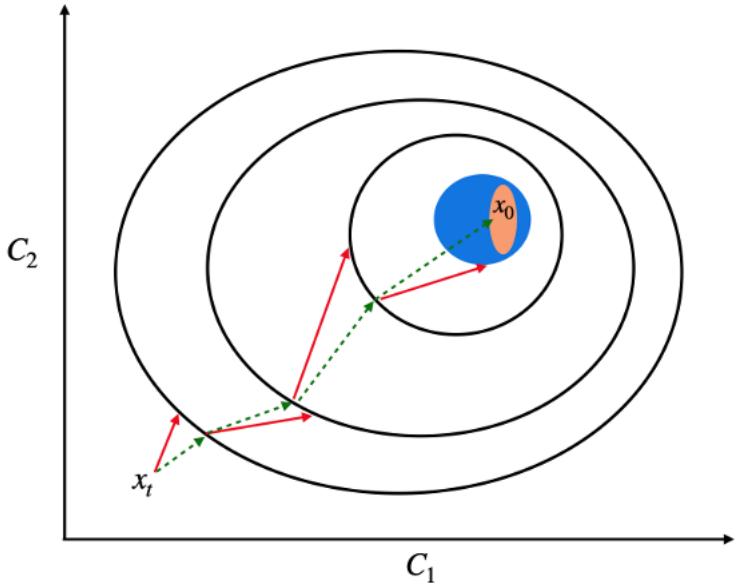


Figure 9: The DPS inpainting process visualized for two representative cells, C_1 and C_2. The axes represent the values the cells can take. The blue area is the precipitation distribution in general to learn and the orange represents the part of the distribution that matches the known values. Every gradient of the sampling step in the reverse process (red) is slightly nudged (green) towards the distribution with the known values by DPS. The figure reduces the distribution to two dimensions whilst the actual distribution to learn is over 1024×1024 dimensions.

reverse process. After a normal sampling step (red) instead of overwriting the values directly, the gradient proposed in that step is slightly nudged into the direction of the subset of the distribution (green).

In Algorithm 2.2.3, the essential structure of the DPS inpainting implementation is given. Most lines are the same as in the sampling, the noise is estimated (line 3) and subtracted (line 4) according to the DDPM paper, some extra noise is sampled (line 5) added (line 6, except for $t = 0$), and the result is clamped (line 10).

Algorithm 1 DPS Inpainting

1: $x_T \sim \mathcal{N}(0, I)$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \hat{\epsilon}_t = \epsilon_\theta(x_t, t)$
4: $\quad \mu_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\hat{\epsilon}_t\right)$
5: $\quad z \sim \mathcal{N}(0, I)$
6: $\quad x_{t-1} = \begin{cases} \mu_t + \sqrt{\beta_t}z, & t > 1 \\ \mu_t, & t = 1 \end{cases}$
7: $\quad \hat{x}_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\hat{\epsilon}_t}{\sqrt{\bar{\alpha}_t}}$
8: $\quad g_t = \nabla_{x_t} \frac{1}{2}\|M \odot (x_{\text{known}} - \hat{x}_0)\|_2^2$
9: $\quad x_{t-1} = x_{t-1} - \lambda g_t$
10: $\quad x_{t-1} \leftarrow \text{clamp}(x_{t-1})$
11: **end for**
12: **return** x_0

The only difference is lines 7-9. Firstly, the final sample $\hat{x}_0$ is estimated (line 7) given the state of the reverse process so far (assuming it being deterministic from there on). Next, the difference between the known cells x_{known} of the stations, selected by a mask M, with the projected final sample is estimated, and subsequently the gradient of the error between them calculated (line 8). This gradient is then multiplied by some factor λ and subtracted from the sample, essentially moving it towards the distribution with realistic rain patches that match the known cells by the factor λ.

2.2.4. Evaluation

Before examining the results, the metrics used are discussed in the following paragraphs. In order to do so, firstly the terminology is defined. Let the ground truth $\mathcal{G}$ be an array of dimensions $T \times W \times H$, where T is the number of samples (here for both hourly and daily model 365), W the width and H the height.

The set of indices of all cells is defined like this:

$$\mathcal{C} = \{\, (t, h, w) \mid t = 1, \ldots, T;\ h = 1, \ldots, H;\ w = 1, \ldots, W \,\} \tag{22}$$

Then $g_c \in \mathcal{G}, c = (t, h, w) \in \mathcal{C}$ is the cell of the ground truth at time step t at cell (h, w) We define $\mathcal{I} \subset \mathcal{C}$ to be the subset of indices in the ground truth where $\forall c \in \mathcal{I} : g_c \in \mathbb{R}$, so the value at the cell is not missing. Similarly, we define the model output $\mathcal{O}$ to have the same dimensions and be indexable in the same way.

Let further $\mu_g = \frac{1}{|\mathcal{I}|}\sum_{c\in\mathcal{I}} g_c$ be the mean of the ground truth, $\mu_o = \frac{1}{|\mathcal{I}|}\sum_{c\in\mathcal{I}} o_c$ be the mean of the model output, $\sigma_g = \sqrt{\sum_{c\in\mathcal{I}} (g_c - \mu_g)^2}$, be the standard deviation

of the ground truth and $\sigma_o = \sqrt{\sum_{c\in\mathcal{I}} (o_c - \mu_o)^2}$ be the standard deviation of the output.

RMSE In training, MSE was used as a loss function to train the weights and biases. For evaluation of pixel to pixel difference between model output and ground truth, RMSE is used here instead. The main advantage is that it allows for an intuitive interpretation - the unit of RMSE score is the same as the one of the input variables. This can be seen via the following formula - the unit is squared, averaged and rooted again.

$$\mathrm{RMSE} = \sqrt{\frac{1}{|\mathcal{I}|} \sum_{c\in\mathcal{I}} (o_c - g_c)^2} \tag{23}$$

Correlation Another aspect related to RMSE is correlation that measures whether both datasets differ from their mean at a cell in the same way. High correlation would indicate that the model predicts rainfall at the same spatial location as the ground truth, even if the magnitude differs. The value ranges from 1, indicating a perfect correlation over 0, showcasing no correlation at all to -1, indicating a perfectly negative correlation.

$$\mathrm{Corr} = \frac{\sum_{c\in\mathcal{I}} (g_c - \mu_g)(o_c - \mu_o)}{\sigma_g\, \sigma_o}. \tag{24}$$

Bias Next to assessing the pixel to pixel similarity, another important factor is assessing whether the model generally over- or underpredicts rain. Here, the bias can be calcualted by averaging the difference between output and ground truth. The values can be positive or negative, indicating over- and underprediction of rain.

$$\mathrm{Bias} = \frac{1}{|\mathcal{I}|} \sum_{c\in\mathcal{I}} (o_c - g_c) \tag{25}$$

SSIM Lastly, it is important to consider, that if a model for example recreates the shape and intensity of a storm perfectly, but is just a few kilometres off, this will yield bad RMSE correlation scores but might be better than a model that gets the shape and intensity wrong but perfectly places the storm. For such and further situations, Wang et al. introduced the Structural SIMilarity index (SSIM) [57].

The index is measured using the following formula, where the first factor measures "luminance", so whether the intensities of the values are similar or not, and contrast/structure by looking at whether there is similar variance.

$$\mathrm{SSIM} = \underbrace{\frac{2\mu_g\mu_o + C_1}{\mu_g^2 + \mu_o^2 + C_1}}_{\text{luminance}} \cdot \underbrace{\frac{2\sigma_{go} + C_2}{\sigma_g^2 + \sigma_o^2 + C_2}}_{\text{contrast \& structure}}, \tag{26}$$

$$\sigma_{go} = \sum_{c \in \mathcal{I}} (g_c - \mu_g)(o_c - \mu_o), \tag{27}$$

and C_1, C_2 are small positive constants added for numerical stability.

Technically, this is more like a combination of the previous metrics. The new idea for SSIM however is that the value is calculated only for a small patch in the output (here 7x7 km^2), so mean and standard variation are calculated for all possible sub-patches of the given size. Subsequently, these values are averaged. The authors show that this way a more intuitive structural similarity between images can be assessed that is less sensitive to spatial shift. The values range from 1 (high structural similarity) to 0 (low structural similarity).

2.2.5. Over- & Underfitting

Diffusion models often train over several hundred epochs, sometimes over a thousand (see e.g. [16, 41]). Like in any deep learning model, a balance needs to be found to avoid over- and underfitting to the training data.

Two methods of dealing with that are the use of *validation data* and *early stopping*. As described in Section 2.1, in this project the available radar data was split into the years 2001-2011 for training, 2012-2017 for validation and 2018 for testing. Hence, if the model is overfitting to the training data over the course of many epochs, validation loss might increase again which can be a way to detect it.

Another option to try to decrease overfitting is to define a maximum period of epochs that a model continues training after loss does not improve any more. This period is called patience and is considered a form of early stopping. Another way to approach this is setting a maximum number of epochs n_{max}, and minimum value δ by which a model needs to have better validation loss at the end of an epoch than in previous epochs. It is considered a form of early stopping even though the model continues to train, and included in major frameworks like Keras and Pytorch Lightning. It is illustrated in Algorithm 2.

Algorithm 2 Training with early stopping via a minimum improvement threshold

Require: Maximum number of epochs $n_{\max}$, minimum improvement threshold δ, untrained model M

Initialize model parameters
$L_{\text{best}} \leftarrow \infty$
$M_{\text{best}} \leftarrow \text{copy}(M)$
for $t = 1$ to $n_{\max}$ **do**
    Train model M for one epoch
    Compute validation loss $L_{\text{val}}(t)$
    if $L_{\text{val}}(t) < L_{\text{best}} - \delta$ **then**
        $L_{\text{best}} \leftarrow L_{\text{val}}(t)$
        $M_{\text{best}} \leftarrow \text{copy}(M)$
    end if
end for
return $M_{\text{best}}, L_{\text{best}}$

The core improvement to patience is, that if model is stuck during gradient descent for several epochs in a plateau, it might still be able to descent to a minima way later. In that case, early stopping is not triggered yet, but also the continued training is only used if there is substantial improvement, defined via δ which can help with overfitting.

This approach was chosen in this project, as it also helps to balance underfitting and resource use: defining a maximum amount of epochs hence accepting that a model might underfit at the end as more training would be infeasible in terms of resource use. For this investigation, the values $n_{max} = 250$ and $\delta = 10^{-4}$ were chosen.

3. Experiment I: Hourly Model

To assess the DDPM, firstly, its reconstruction of the ground truth used for training via station data inpainting is assessed in this experiment. The training of the model is described in Section 3.1. Next, the inpainting results are presented in Section 3.2. Lastly, the results are analysed in an initial experiment discussion in Section 3.3.

3.1. Training

The training of the model is showcased via its learning curve and sampling results shown in the paragraphs below.

Learning Curve In Fig. 10 the learning curve of the hourly model for both training and validation loss (MSE) is portrayed. It follows an expected steeper descent in the beginning and slower in the end, and (mostly) a higher validation loss than training

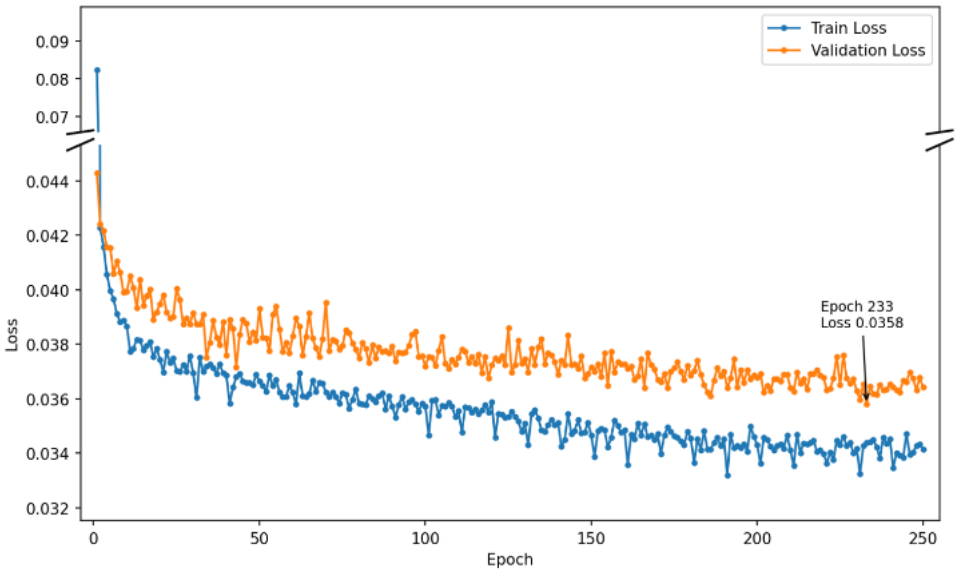


Figure 10: Training of the hourly model over 250 epochs, including training and validation loss.

loss. The best model was achieved in Epoch 233 with validation loss 0.035797 and had sufficient improvement to the previous best of 0.035957 in Epoch 231.

Sampling To test the model performance, samples can be generated, based on completely random noise (so no inpainting). In Fig. 11 the process of sampling over various epochs (10, 50, 150 and 233, the final best epoch) is showcased, compared to samples from the RADKLIM dataset.

Subjectively, the difference does not appear to be big apart from seemingly bigger connected rain-fields in later epochs. A more formal approach is to compare samples over various epochs with statistical tools. Here, histograms and *radially-average power spectral density (RAPSD)* curves are used to compare *RADKLIM* with generated samples. The results are showcase in Fig. 12.

RAPSD curves use Fourier transforms to analyse the power (variability) at different spatial frequencies (signal change over space) [14] that have been used as analysis tools in similar research [47, 19].

For example, high power at a low frequency indicates high variability in bigger structures in an image. If the RAPSD curve of generated and real values look similar, this indicates that the model learns a distribution that plots similarly big rainfall structures and variability within them.

For the RAPSD curves, an improvement can be seen over the course of the trainings, with the real and generated curves becoming closer. The histogram values for the generated samples do not differ much, apart from an unusual spike around 30 mm / hour at epoch 150. Overall, training appears to be stable.

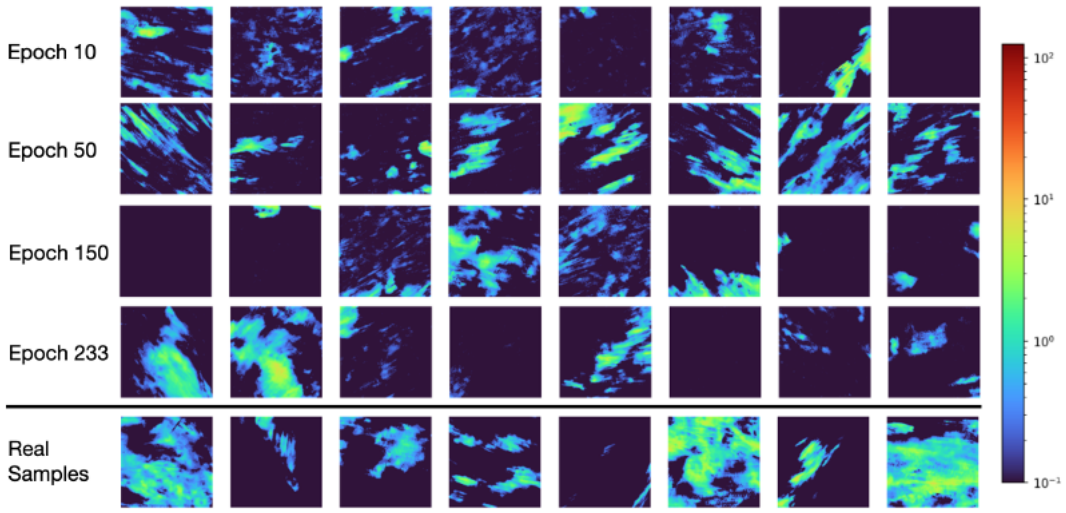


Figure 11: Samples produced by the hourly model at different epochs, compared to samples from the *RADKLIM* hourly dataset. Sample distribution varies a lot so a small subset provides just an impression.

3.2. Results

To examine the model results, first the inpainting is qualitatively checked and compared against the station data input and the ground truth. Next, the results are evaluated quantitively with the metrics described in Section 2.2.4.

3.2.1. Inpainting

For the actual inpainting task, first the λ parameter described in Section 2.2.3 needed to be optimized. In order to do so, a 0.01% mask was applied to the radar data, and the model was used to inpaint the masked data with different configurations of λ. The output was compared with the original and the value with the lowest MSE being found at $\lambda = 0.02$, see Fig. 13.

Subsequently, the data was inpainted. As the hourly model had $365 \cdot 24 = 8760$ and inpainting 1024x1024 patches took substantial resources, only a subset of 365 time steps was inpainted. From every day an hour was selected at random under the constraint that in every month every hour of the day was chosen once. This was conducted to ensure that the results are stable over the whole year and all-day.

The results are portrayed in Fig. 14. It is evident that the inpainted patches are not identical to the radar data, but the general shapes and intensities seem to be preserved.

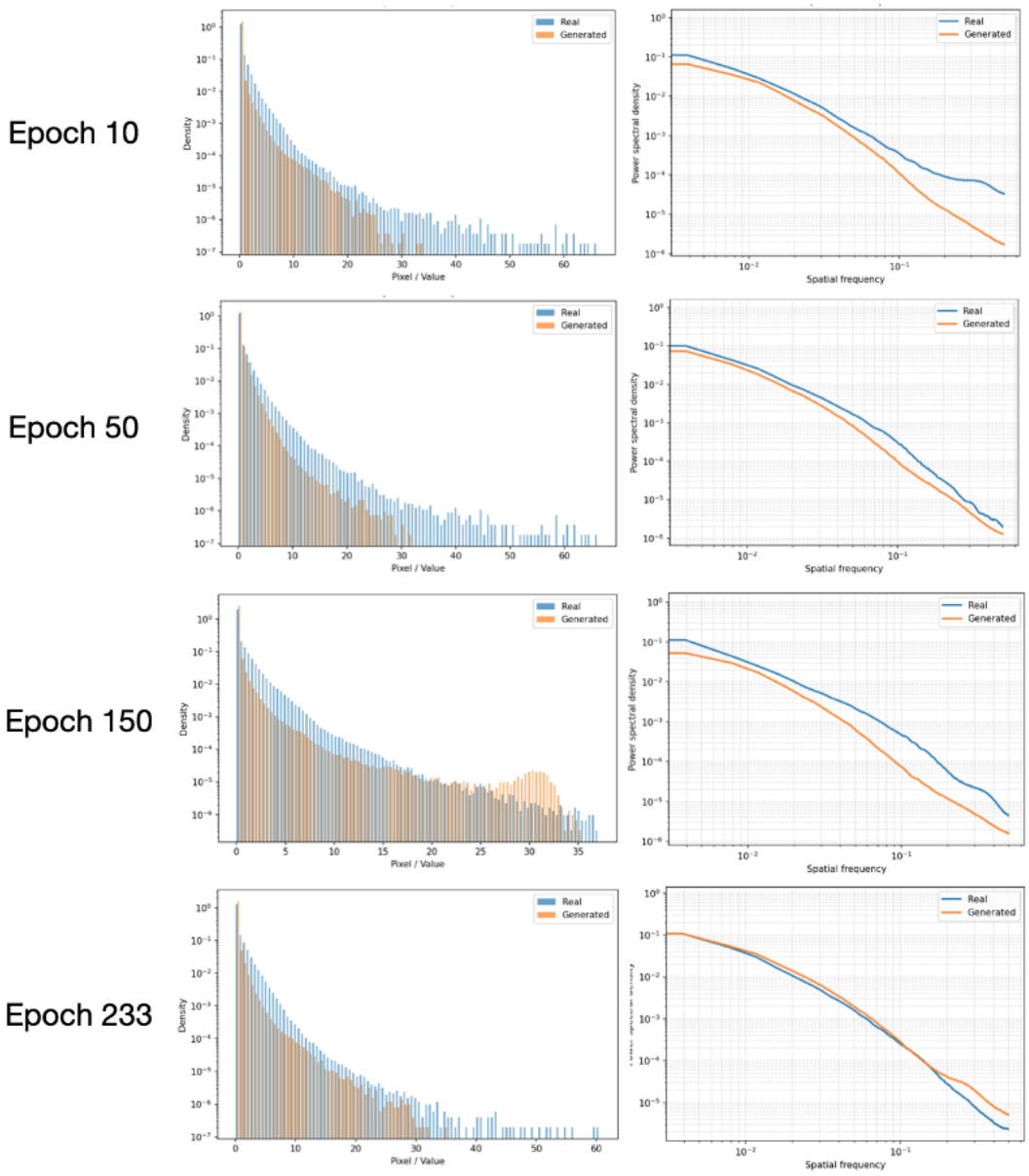


Figure 12: Histograms and RAPSD plots of model samples at different epochs for radar/real (blue) and sampled/generated (orange) rain patches. An improvement can be seen over the course of training in the RAPSD curves.

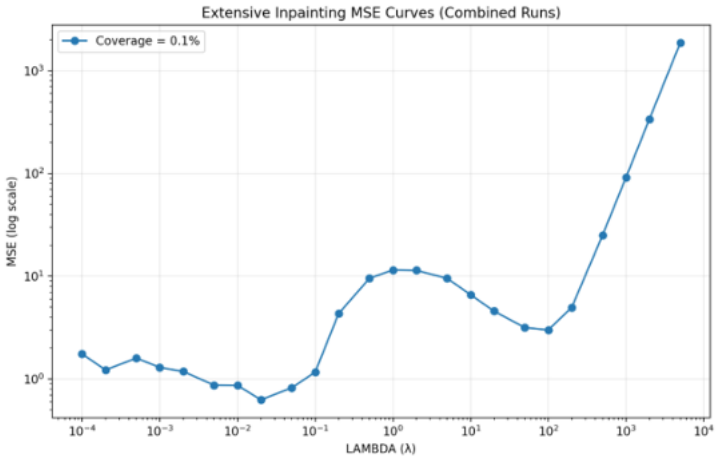


Figure 13: Test of different lambda values in inpainting. A random mask of 0.01 % was applied to 20 256x256 patches in the *RADKLIM* dataset and MSE of the inpainted patch in comparison to the original was measured. Best error was reached at $\lambda = 0.02$.

3.2.2. Metrics

The results for the hourly model for the four described metrics RMSE, Correlation, Bias and SSIM described in Section 2.2.4 are shown in Table 5. The RMSE is 0.5772 mm/hour with a mean (over all the inpainted time steps) of 0.3359 and a 40% higher standard deviation. A slight positive correlation of 0.30 can be observed. The inpainted stations exhibit very little bias in relation to the ground truth, slightly overestimating the values and the structural similarity appears to be high with a SSIM score of 0.88.

Metric	**Value**	**Mean (over time**
Std (over time)		
RMSE	0.5772	0.3359
0.4696		
Correlation	0.2965	0.2182
0.1995		
Bias	0.0088	0.0087
0.0334		
SSIM	0.8799	0.8799
0.0949		

Table 5: Summary of evaluation metrics for the inpainted fields of the hourly model.

Next, model performance over the daily and yearly cycle is evaluated in order to see if the model misperforms at specific times of day or points in the year or rain

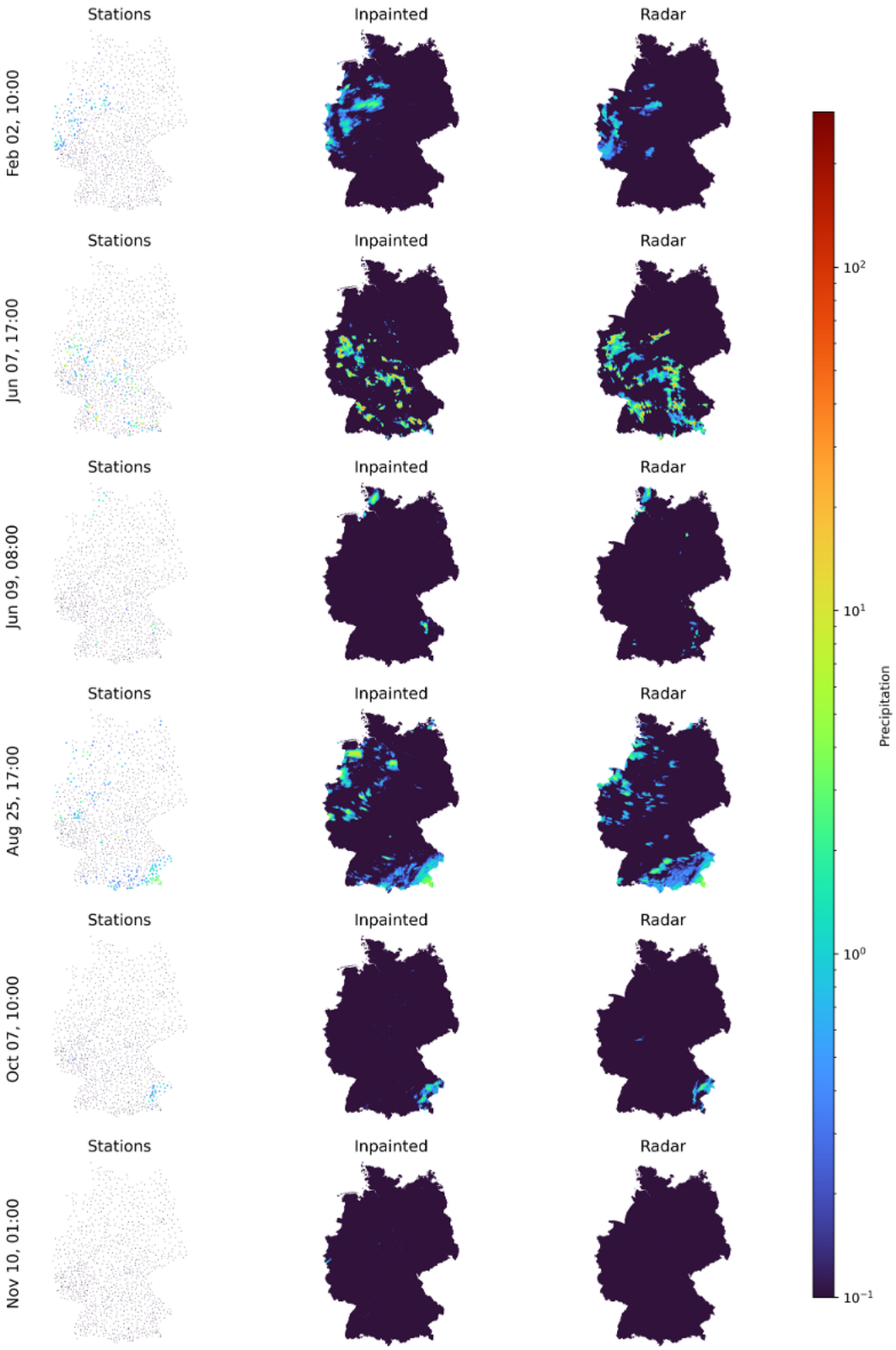


Figure 14: Inpainted hourly station data at six random time steps, compared with the ground truth radar data. The station data points have been magnified to make them visible in comparison in this and all following station figures. Plots are shown on a log scale to make low precipitation visible.

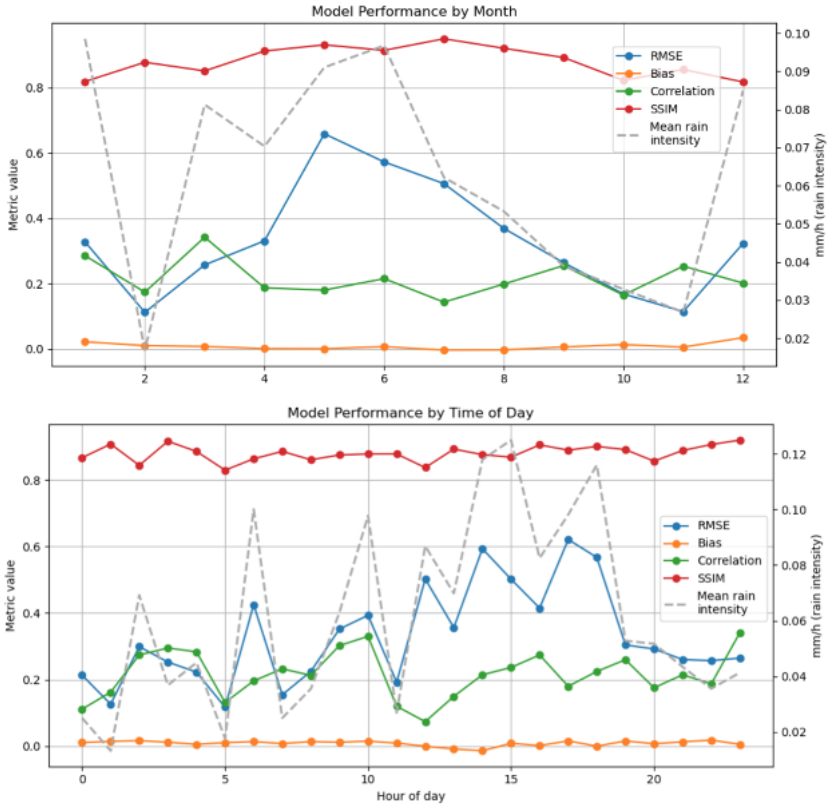


Figure 15: Model performance in the annual (top) and diurnal (bottom) cycle for the four metrics. Rain intensity is added for reference on a different scale, to make trends comparable. Whilst Bias, and SSIM seem fairly consistent, correlation and RMSE vary more, the latter seeming linked to rain intensity

intensities. The results can be examined in Fig. 15. .

Average rain intensity (in mm/h) was also added for reference. A clear seasonal pattern can be seen for the year 2018, with most rain occurring in the winter months (Jan, Dec) and in spring (March through June) and a drier summer.

It is noticeable, the the RMSE is often much higher than the average rain intensities. Bias remains very low and SSIM very high throughout the day and year. Correlation appears to be mostly independent of rain intensity.

3.3. Experiment Discussion

In this section, the results of the first experiment, the hourly model, are analysed. The output is discussed in relation to its architecture, strengths and weaknesses are examined and ideas for improvement proposed.

Strengths The model seems to have learned well over time, potentially an even longer run might have produced even better results as validation loss still appeared to descent in the last epochs. Subjectively, the samples produced seem realistic and similar to ground truth samples. Hence, the learning of the data distribution is at least somewhat successful.

The bias is consistently very low, also in relation to the rain intensity. In most months, the bias is slightly positive and only shortly after noon it is a briefly negative. This shows that the model slightly overestimates precipitation magnitude, but not substantially.

The hourly model has good SSIM scores and the RAPSD curve is very close to the ground truth values, even though the latter must be viewed with caution as it is only based on 128 256x256 patches sampled from the model. This indicates that the model output matches the learned distribution structurally which might be due to the more selective importance sampling in the hourly model, where only 2% of patches were chosen, favouring high-precipitation areas (see Section 2.1.2).

Weaknesses Without comparison, the RMSE of 0.58 mm/h is hard to interpret. However, it is noticeable that it rises and falls with rain intensity. The high standard deviation of 0.47 also indicates that when there is precipitation, the error gets even worse.

Another drawback of the model appears to be the small positive correlation. Both suggest that the model might output unbiased and structurally plausible rain fields, but does so a few kilometres away.

Whilst the histogram improved over training, across all epochs radar values were higher than the generated values, whilst the generated ones had a higher peak at 0. When examining further, the 99.9th percentile of the ground truth (5.9mm/h) and the model output (6.1 mm/h) is similar, but the maximum value in the ground truth (77.4 mm/h) is much higher than the one from the output (39.4 mm/h).

This might be due to the scaled distribution never having values < -0.5 whilst the positive values go up to around 6. However, during inpainting we start with Gaussian noise that should have a similar positive and negative range.

Ideas The spatial shift of clouds could be improved by considering geographic information. Rainfall might often occur on the side of a mountain, close to the sea or in a specific area at a specific time of year.

The model as trained only works with one input channel but could easily be extended to hold information about altitude, longitude, latitude and embedded temporal information. Further, the overestimated zeroes could be encountered with a more aggressive clamping approach.

Another option to improve the location is to include geographical coordinates in training, as done in the REGNIE method (see Section 1.4.1), for example via an extra channel.

4. Experiment II: Daily Model - HYRAS comparison

After assessing the hourly model trained on the RADKLIM dataset, the first comparison to a statistical interpolation approach, the REGNIE method described in Section 1.4 can be discussed.

As HYRAS, the resulting dataset is only available at a daily resolution, the main difference is that the hourly radar training data was aggregated to days and used in a new model, with a separate importance sampling scheme resulting in about 50 % of the patches being used.

Again, the experiment presentation is split into training (Section 4.1), results (Section 4.2) and experiment discussion (Section 4.3).

4.1. Training

As mentioned in Section 2.1.2, the hourly model was trained with stricter importance sampling, leading to only 2% of available patches used. In the following the model, learning process is discussed as well as the model's samples throughout training.

Learning Similarly to the hourly model, the daily model's training and validation loss falls rapidly in the first epochs but seems to converge more quickly to minima during gradient descent as shown in Fig. 16.

Here, the overfitting prevention described in Section 2.2.5 is used. Whilst in 250 epochs the model reached its lowest validation loss at epoch 236 with a loss of 0.02193, the improvement does not exceed the threshold $\delta = 0.0001$ to the former best model in epoch 181 at a loss of 0.02194. Therefore, the model 181 was chosen as the best model due to the possibility that the model could have overfitted to the training data in the epochs 181-236.

Sampling The samples produced during training are shown in Fig. 17. Again, only eight samples show a small subsection of learned distribution. It seems that the model learns subjectively, already realistically looking at rain patches early on.

However, in Fig. 18 we can see that the RAPSD curves become more similar over the course of training. However, whilst the histogram similarity between real (radar) and generated (model) values seems to overlap nearly perfectly in Epoch 100, the results get worse in epoch 181.

Specifically, the model seems to overpredict very low (and partly very high) rain intensities again, and appears also to underestimate rainfalls between 5 and 80 mm/day.

4.2. Results

First, some examples of the inpainting of the model next to the station data and the ground truth are presented, in this case including the HYRAS reconstruction,

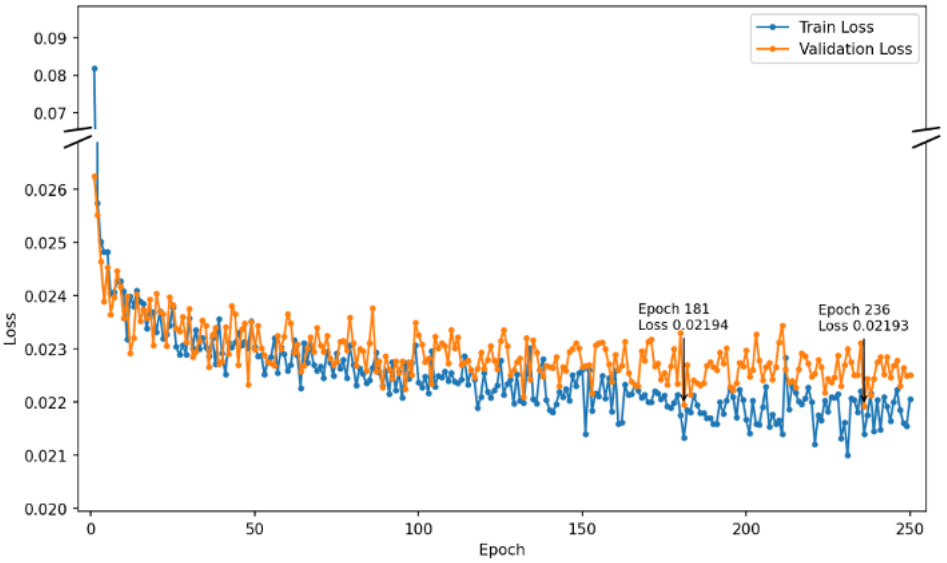


Figure 16: Training of the daily model - significant improvement of validation loss seems to stop after epoch 181.

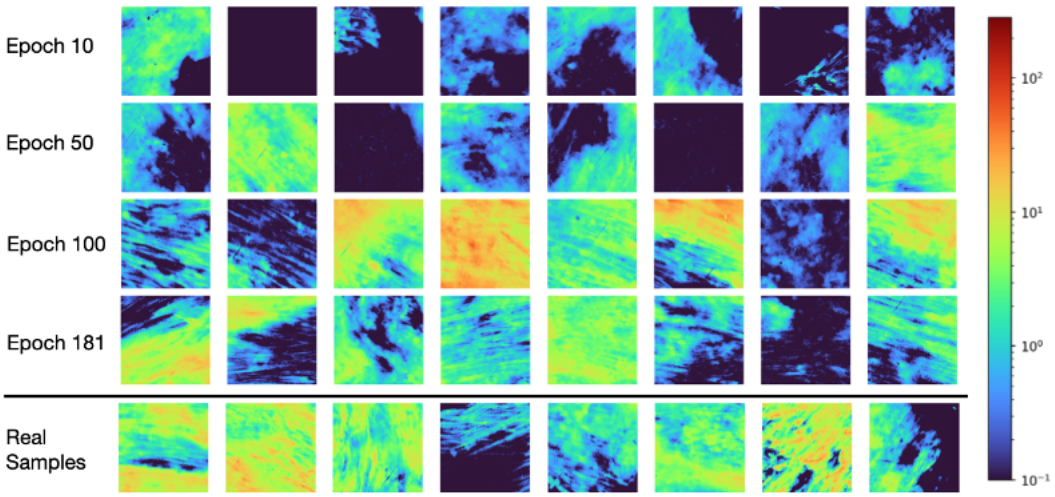


Figure 17: Samples produced by the daily model at different epochs compared to real samples from the ground truth RADKLIM dataset.

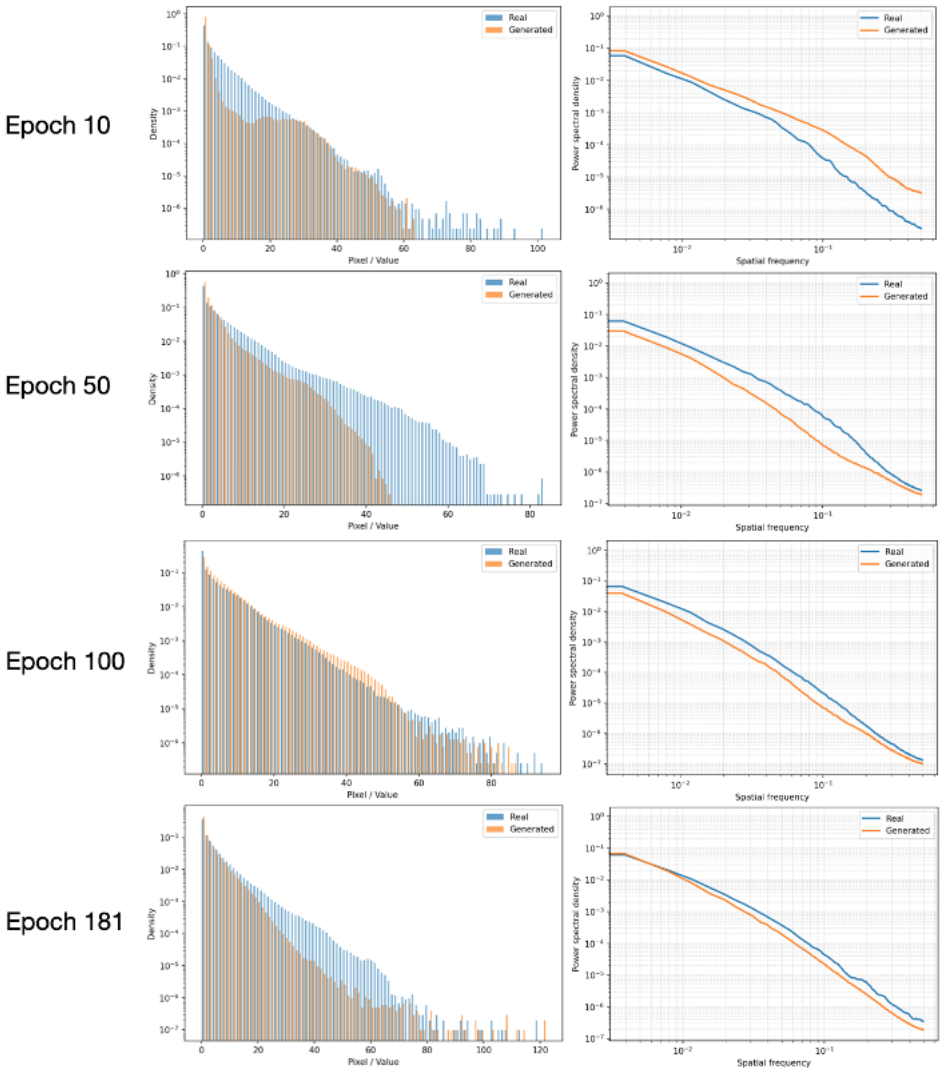


Figure 18: Histograms and RAPSD plots of daily model samples at different epochs for radar/real (blue) and sampled/generated (orange) rain patches. The histogram similarity seems to be much better at epoch 100 than epoch 181.

followed by a discussion of the model output performance.

4.2.1. Inpainting

The inpainting of the daily station data is shown in Fig. 19, compared to the station data itself, the ground truth radar data, and the HYRAS interpolation for six randomly chosen days throughout the year.

In general, the inpainted and HYRAS data seem most similar as they are based on the same data. On some days, the HYRAS dataset seems slightly more blurry than the other complete datasets.

4.2.2. Metrics

As the model inpainting subjectively seems to have performed well, the performance is tested using the metrics described in Section 2.2.4 in the following paragraphs.

The main results of the model output compared to the ground truth are shown in Table 6, together with the equivalent HYRAS/ground truth comparison.

Metric	Model Output	HYRAS
RMSE	2.8114	2.4251
Correlation	0.7760	0.8299
Bias	0.1180	0.0153
SSIM	0.7466	0.8558

Table 6: Evaluation metrics for daily inpainted precipitation fields: model output compared to HYRAS. NaN values are ignored.

Whilst a positive correlation of 0.78 can be seen and some structural similarity is observed with a SSIM score of 0.75, HYRAS outperforms the daily model in all metrics. This includes RMSE which 0.4 mm/day lower in the HYRAS daily model as well as roughly a tenth of the bias the DDPM daily model has. Interestingly, both models seem to slightly over predict the rainfall in relation to the ground truth.

Furthermore, it is interesting to note that in most months, the RMSE exceeds the mean rain intensity, both for HYRAS and the DDPM inpainting output.

In Fig. 20 the metrics are examined over the months of the year. It can be seen that HYRAS consistently outperforms the DDPM model.

The only category where the DDPM model is slightly better is in the BIAS in the spring and summer months April through August, as both models slightly underestimate the ground truth values on average in those months and the DDPM seems to in general predict slightly higher values than HYRAS.

Next, the power spectral densities of the ground truth, HYRAS and the DDPM model output are compared. The results are presented in Fig. 21. For very low frequencies below 10^{-2} the models mostly match the ground truth.

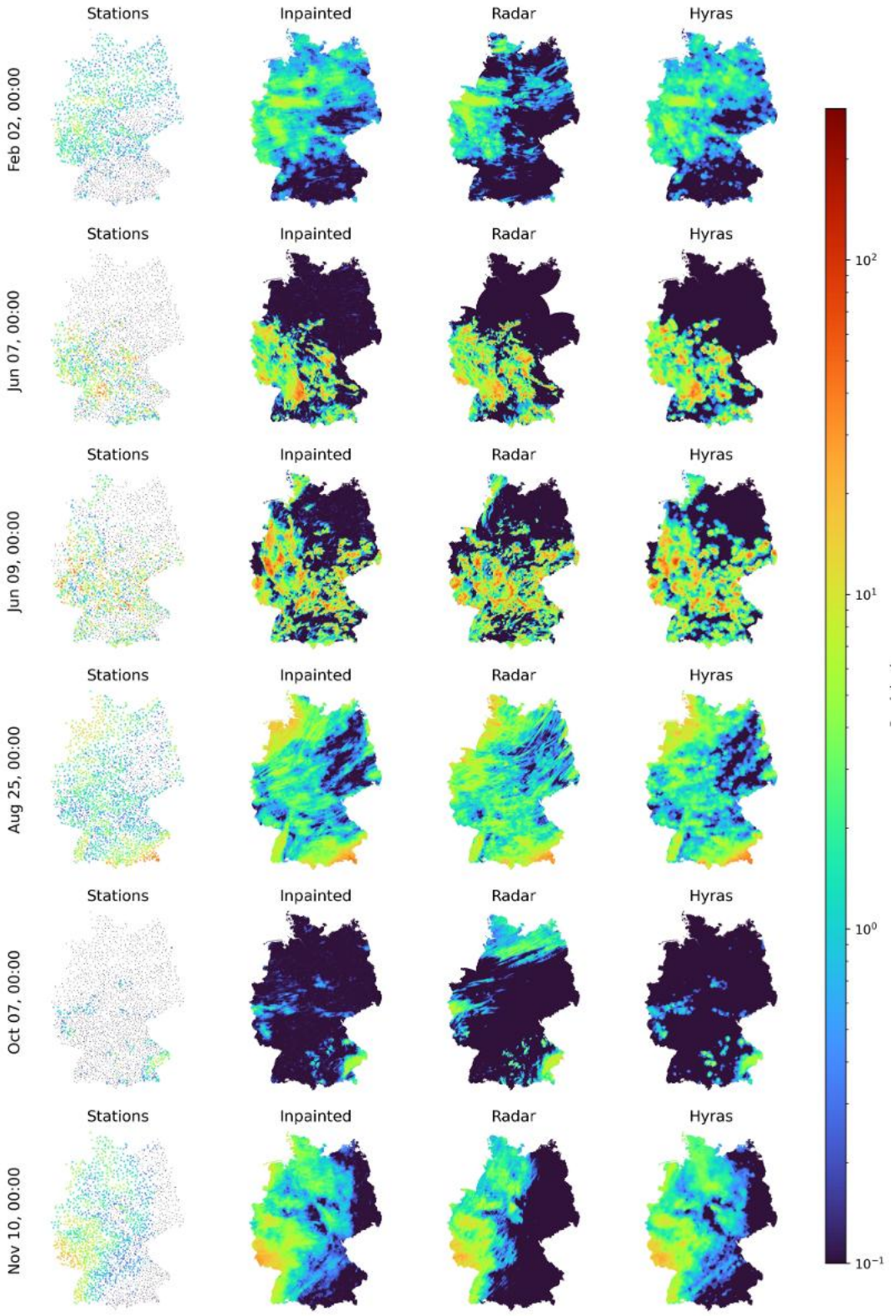


Figure 19: Precipitation at six randomly chosen days of the year (rows) for the station data itself (white areas represent missing values), the inpainted station data, the ground truth radar data and the HYRAS interpolation.

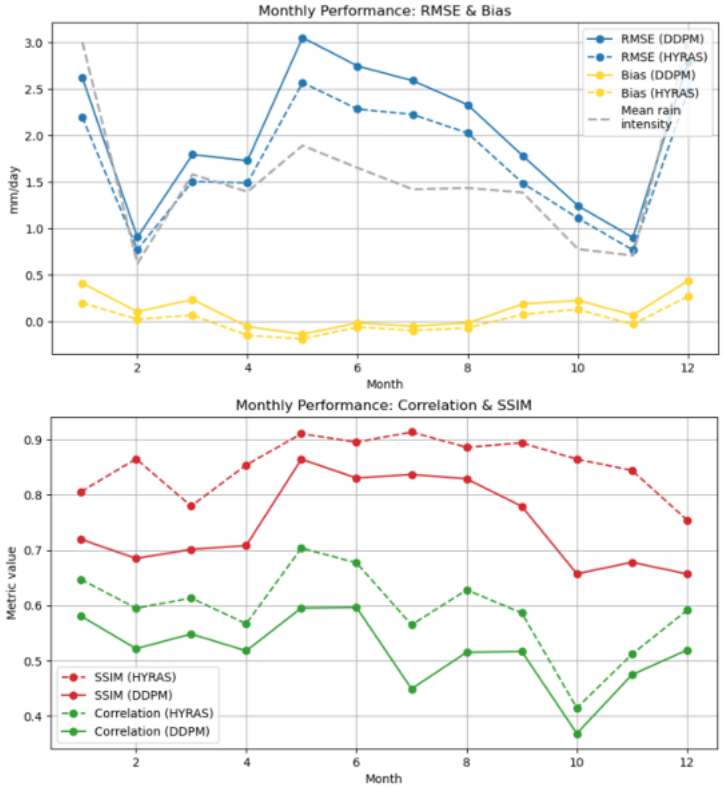


Figure 20: HYRAS compared to the daily DDPM model output over the course of the year: RMSE & Bias (top) and SSIM & Correlation (bottom). As RMSE and Bias as well as rain intensities are measured in mm/day.

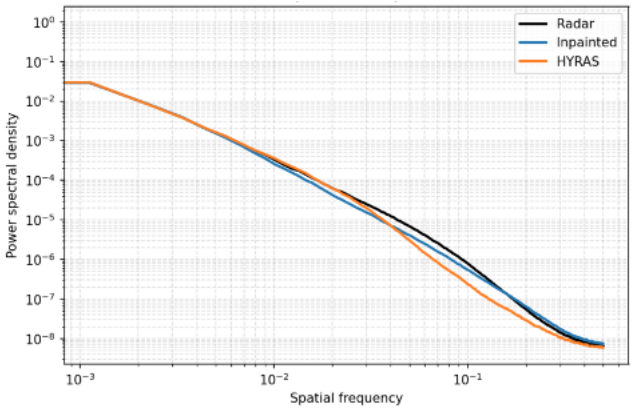


Figure 21: Radially averaged power spectral densities (RAPSD) of the ground truth (radar), DDPM (Inpainted) and HYRAS datasets. The RAPSD curves were normalized (sum to one) here to make them comparable.

For medium frequencies slightly above that, the DDPM inpainting has lower energy than the other two. This shows it slightly underestimates the variability of medium sized structures, such as potentially smaller rain bands around a mountain range.

For higher frequencies, i.e. smaller structures such as little regional showers, the DDPM Inpainting matches the ground truth slightly better, whilst HYRAS predicts less variability or 'power'.

As the HYRAS dataset clearly matched the ground truth better than the DDPM output in most regards, three further checks were conducted to test the DDPM model performance:

1. ***Other Epochs***: As the histogram of the daily model seemed worse in epoch 181 than in epoch 100, and the early stopping might have disrupted potential further learning, the station data was inpainted for the models at epochs 100 and 220 as well.

2. ***Monte-Carlo Average***: Deterministic models might be better than a probabilistic model like a DDPM but produce blurrier results. Effectively, the model is learning the distribution but can create loads of possible inpaintings of a masked input based on different initial noise fields drawn in the reverse diffusion. Hence, a *'Monte-Carlo'* mode was built, where the DDPM performed the inpainting 10 times and the results were averaged. This was only conducted for 36 time steps (every tenth day of the year) to save computing power as this approach is more resource intensive.

   There are various Monte-Carlo methods all based on repeatedly sampling from a distribution [5], the most basic of which (sampling and then averaging) is used here.

3. ***Hourly Aggregate***: As the hourly model had been trained anyway, it was used to inpaint all 24 hours of selected days and then aggregated to obtain a daily inpainting. One random day each month was chosen, again to save computing resources.

The results are shown in Table 7. Here, the metric RAPSD Difference, indicating the total difference between the values of the normalised model and ground truth curve, is added as an indicator of power spectral density similarity.

Whilst it shows that epoch 181 was worse than the other two epochs examined in the categories RMSE, Bias and SSIM, still none of the models at different epochs outperform HYRAS in any category.

However, the inpaintings of the Monte Carlo averaging perform much better than those of the not averaged models and get much closer to the HYRAS metric results. Bias is even slightly better than HYRAS, the values are much higher than in the epoch comparison which is likely due to the days chosen at random having higher rain intensities.

Table 7: Comparison of Inpainted Models and HYRAS across Different Evaluations. The better value is highlighted to compare performance more easily. HYRAS has slightly different values in all comparisons as only a subset of the time steps was used to save computing resources.

Model	RMSE	Correlation	Bias	SSIM	RAPSD Diff.
Daily Model vs HYRAS					
HYRAS (daily)	**2.4251**	**0.8299**	**0.0153**	**0.8558**	**0.001831**
Epoch 100	2.7915	0.7784	0.1127	0.7571	0.003929
Epoch 181	2.8114	0.7760	0.1180	0.7466	0.001972
Epoch 220	2.7724	0.7789	0.0801	0.7685	0.001902
Daily Monte Carlo vs HYRAS					
HYRAS (MC Days)	**2.4643**	**0.8044**	0.1668	**0.8614**	**0.004734**
Monte Carlo	2.4952	0.7805	**0.1627**	0.7671	0.006647
Hourly Aggregated vs HYRAS					
HYRAS (HA Days)	2.9085	0.8756	-0.5590	**0.8417**	0.006366
Hourly Agg.	**2.2354**	**0.9275**	**0.3284**	0.7627	**0.004338**

The similarity between the results is shown in Fig. 22. RMSE is the same in drier months and HYRAS is only slightly better than DDPM in the wetter months with regard to RMSE. As mentioned, Bias is slightly better in the Monte Carlo daily DDPM than in HYRAS, and the model does not consistently estimate higher values than HYRAS like it did before.

The hourly aggregate model performs better than HYRAS in every category apart from SSIM. However, it is important to note that this model had hourly station data available, so a higher temporal resolution than HYRAS, which only used daily station data.

4.3. Experiment Discussion

In summary, the results of the daily DDPM comparison to HYRAS point to a better performance of HYRAS. Even though the Monte-Carlo DDPM version achieved comparable results, this shows that an initial implementation of a machine learning model did not outperform a traditional statistical model.

This section is again separated into strengths and weaknesses of the daily DDPM model followed by a discussion of ideas for improvement.

Strengths The baseline (epoch 181, no modifications) model achieved some positive correlation and structural similarity to the *RADKLIM* dataset. RAPSD curve similarity was comparable to the one of HYRAS, and in general very close to the radar data.

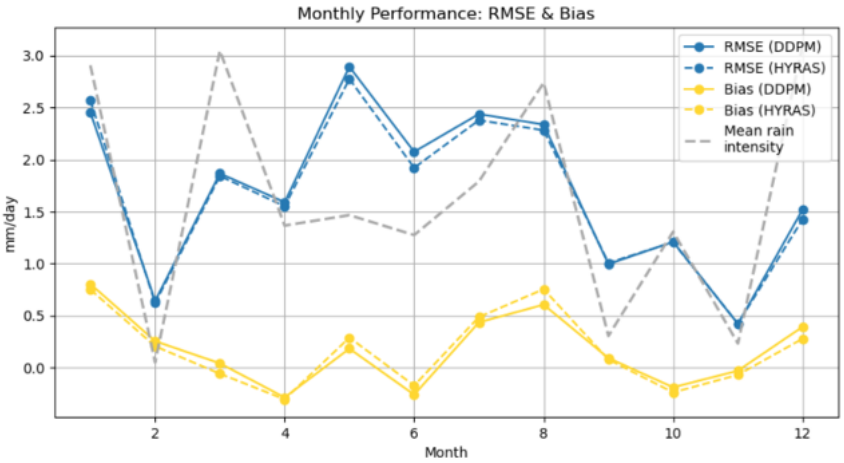


Figure 22: RMSE and Bias comparison of both HYRAS and the Monte-Carlo DDPM model output against the ground truth as well as average rain intensities.

This shows that the model in general learned a realistic-looking distribution, which can be confirmed by subjective interpretation of the samples produced.

The Monte Carlo daily model achieved comparable results to HYRAS in all categories except SSIM and RAPSD Difference, the latter potentially being due to the specific time steps chosen.

As this was just a master's thesis project compared to the much evolved and specialized HYRAS approach, this shows that there is a high potential of DDPMs to surpass HYRAS in the future.

Weaknesses Given the prevalent structural similarity but low bias exhibited by both the DDPM and especially the HYRAS model in the early summer months, this indicates that both approaches recreate similar and realistic rain patches but are likely often slightly misplaced from the ground truth. As SSIM was measured in 7 km^2 patches, this misplacement of rainy cells was off by more than a few kilometres.

As average rain intensity was in some months in a similar order as the RMSE of both HYRAS and the DDPM, both results are not ideal. However, the DDPMs RMSE in the unmodified version was 15 % worse than the HYRAS comparison.

Even if other metrics were slightly better than HYRAS, this deems the DDPM output less usable than HYRAS for now when using the output datasets for location-specific precipitation information. This, however, is often the main purpose of such missing value estimators.

Ideas The daily model was built on about 50% of available radar data through importance sampling. Utilising all of it, even if it uses less structural information could be a good step to test the value of the importance sampling scheme.

It was pointed out that the hourly aggregate model is, while better, not a fair comparison to the HYRAS model due to it having access to more station data than HYRAS in training.

However, it being immediately much better than the daily DDPM model and even better than the HYRAS model showcases the unused information in the HYRAS model. This shows that future iterations of HYRAS have a high potential to significantly improve if they better utilised this data.

HYRAS, on the other hand, had more auxiliary data than the DDPM model available - it used coordinates and altitude information. As previously mentioned, this could significantly improve the DDPM performance in a still fair comparison when used with daily radar data, especially in the Monte-Carlo model.

5. Experiment III: Sparse Hourly Model - U-Net comparison

For the last experiment, the DDPM was compared to a similar model that uses the U-Net Architecture. It is based on a conference poster by Filippou et al. [13] and a master's thesis by the first author and was discussed in Section 1.4.

The key difference in the U-Net inpainting experiment is that it was only done for 76 stations (the exact 76 stations described in Section 2.1 that have hourly station data going back to 1951) resulting in a station coverage of only 0.02 % of Germany on a 1 x 1 km^2 resolution.

The reason for this is that it is interesting to test how stable the reconstruction of climate data will be going back decades with a realistic coverage.

In this experiment, the same model was used as described in Experiment 1 (Section 3), but it was inpainted using the exact same 76 stations as in the U-Net Model. It will be referred to from here on as the Sparse DDPM.

Specifically, four different modes are used:

1. Standard Sparse DDPM using one random hour each day as described in Section 3

2. Sparse DDPM with Monte Carlo Averaging (mean of 10 inpaintings) for a random hour every tenth day of the year

3. Daily Sparse DDPM model by aggregating the full 24 hours of a random day each month (as described in Section 4)

4. Daily Sparse DDPM with Monte Carlo averaging (mean of 5 inpaintings) for the full 24 hours of one random day each month

This section begins with a note on exactly how the output of the DDPM can be compared to the results of the Filippou et al. U-Net in Section 5.1, followed by the actual results in Section 5.2 and concludes with a discussion of these in Section 5.3.

5.1. Filippou Comparison

In the master's thesis, the author runs ten different U-Net based models using the architecture described in Section 1.4. They differ in regard to how much secondary information such as wind and temperature and precipitation station data from previous time steps is included in additional initial channels next to the station precipitation channel.

As no additional information was used in the DDPM, the results are compared to run number 1 in this thesis which does not use any additional information, unless otherwise stated. Results from other runs are amended where appropriate for context.

In contrast to the inpainting conducted in this thesis, the U-Net model output includes all time steps of the year 2018, (every hour) whilst for all modes described above, only a subset was used.

As a further difference, both results were mapped to the HYRAS grid to make all three comparable. Whilst the U-Net output was also remapped using nearest neighbour mapping, the implementation could have been slightly different to the one used here. It is important to take these aspects into account when analysing the results.

Filippou saw the *RADKLIM* dataset as the ground truth in the model as well, but also compared the model's results to HYRAS directly. Whilst similarity to HYRAS is less informative than comparison to radar data as it is a secondary data source, radar data has various quality issues (as described in Section 2.1 and HYRAS was developed and calibrated by industry experts.

To establish a proper comparison, the author was contacted to provide the actual output of the U-Net. Unfortunately, these results could not be retrieved as they now work at a different place. Only a previous output was still found at the University of Hamburg but it turned out to perform much worse than the one described in the thesis.

Therefore, the comparison to the U-Net is conducted by comparing the main metrics discussed in the thesis itself. These are as follows:

1. **RMSE** - the cell to cell root mean squared error was calculated the same way in both the U-Net and the DDPM. The U-Net also offers a month-specific output.

2. **Correlation** The U-Net also assessed the Pearson Correlation Coefficient equivalently to the formula used here. However, the results are only provided in a monthly figure, making the comparison less exact.

3. **Temporal Correlation Maps** Next to spatial correlation as described above, the U-Net model output's temporal correlation is also assessed via *temporal correlation maps*. As these results are only provided in the form of maps, they can only be graphically compared which is avoided here due to it being highly subjective (unless using specialized frameworks).

4. **Total precipitation Difference** This metric is similar to bias and sums the difference between ground truth and model. The formula is given in Equation 28, where again T, W, H represent the range of time steps (t), height (h) and width (w), of all ground truth $g_{t,h,w}$ and output $o_{t,h,w}$ cells respectively.

$$\text{PRdiff} = \sum_{t=0}^{T}\sum_{i=0}^{H}\sum_{j=0}^{W} g_{t,h,w} - \sum_{t=0}^{T}\sum_{i=0}^{H}\sum_{j=0}^{W} o_{t,h,w} \tag{28}$$

5.2. Results

Before looking at the results of the comparison to the U-Net, first the output is examined in general in Section 5.2.1. Next, the U-Net comparison is conducted in Section 5.2.2. Lastly, a note on the similarity of both models to the HYRAS dataset based on the REGNIE method is written in Section 5.2.3.

5.2.1. General

The model output for the inpainting of the 76 stations can be seen in Fig. 23. It is evident that the ground truth (Radar) and DDPM output (Inpainted) differ significantly. The ground truth seems to have in general finer structures, whilst the DDPM estimates bigger connected precipitation fields.

Looking at the results of the metrics, surprisingly, the sparse hourly model (with only 76 stations) performs better than the baseline hourly model from the first experiment (Section 3) that used all available stations in 2018, at least in terms of RMSE, which might be due to the specific subset of time steps chosen.

Bias of both models compared to the ground truth radar data is similar, and correlation and structural similarity (measured again via SSIM) is worse in the sparse model than in the baseline model.

Especially spatial correlation of the sparse model is worse here, with a Pearson Coefficient of 0.206 compared to 0.297 in the baseline model.

Table 8: Evaluation metrics for the sparse hourly model compared to the Hourly baseline. Mean and standard deviation are computed over all 365 inpainted time steps.

Metric	Sparse Model	Mean	Std	Baseline Model
RMSE	0.5301	0.3484	0.3997	0.5772
Correlation	0.2059	0.1111	0.1410	0.2965
Bias	0.0087	0.0087	0.0650	0.0088
SSIM	0.8338	0.8338	0.1011	0.8799

For completeness, the performance of all metrics over the course of the year and day is portrayed in Fig. 24. It shows that RMSE is again dependent on rain intensity, and that rain intensity, is on average much lower than the root mean squared error.

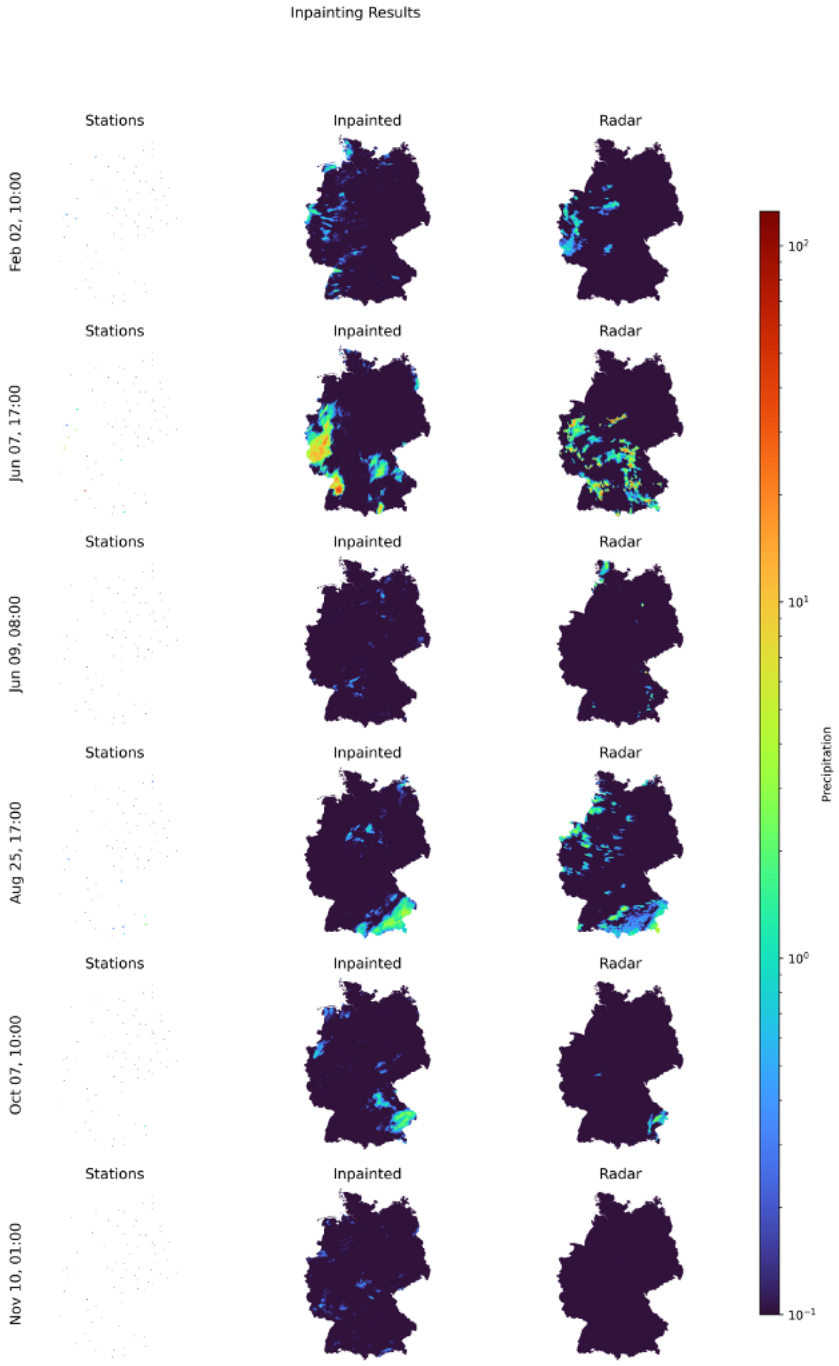


Figure 23: The inpainted hourly model with only data from the 76 stations. Input data is evidently very sparse, showcasing the difficulty of the missing value estimation task.

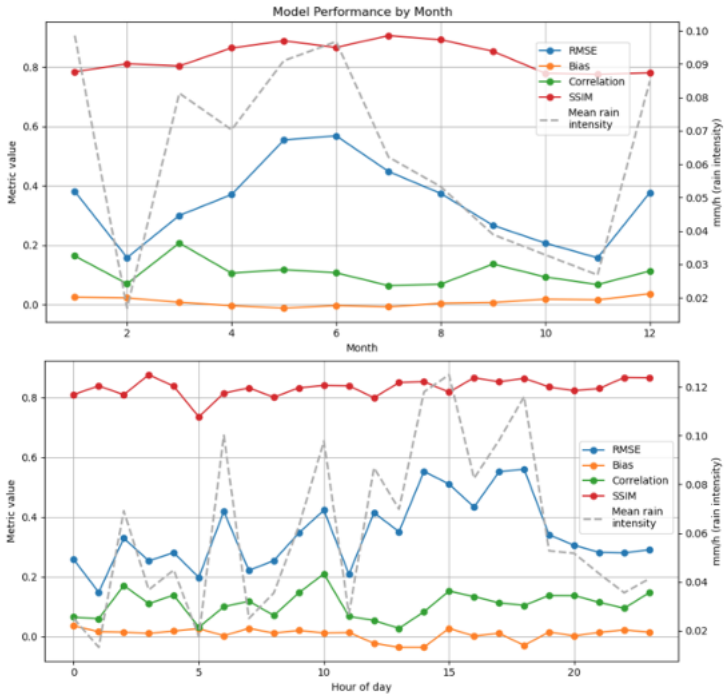


Figure 24: Sparse hourly model over the course of the year (top) and day (below) for RMSE, Correlation, Bias and SSIM. Average rain intensity is amended. Notably, it uses a different scale (on the right) than RMSE and bias to make trends comparable.

Correlation falls roughly between 0.1 and 0.2 over the course of the year, in line with the correlation mean of 0.111 shown in Table 8. Bias seems mostly at a stable low throughout months and hours and SSIM has a slightly higher value in the summer months.

5.2.2. U-Net comparison

For the U-Net comparison, the three metrics chosen, *RMSE*, *Correlation* and *Total Difference* are presented separately.

RMSE The results of the RMSE comparison are shown in Table 9. Next to the sparse U-Net and DDPM model results themselves some model variants are added for reference, for the DDPM the ones described above and for the U-Net the best performing U-Net run (in terms of RMSE). This was run number 5 that utilized wind data in an extra channel next to the precipitation channel.

The U-Net clearly performs much better than the (non-aggregated) DDPM, with the sparse DDPM's RMSE being more than three times higher. Whilst the aggre-

Table 9: RMSE of U-Net and Sparse DDPM model variants against the radar data at hourly and daily (for the aggregated DDPM models) resolution, given in mm/h.

Model type	U-Net	U-Net Run 5	DDPM	DDPM Monte Carlo	DDPM Aggregated	DDPM Aggregated/ Monte Carlo
RMSE	0.1611	0.1594	0.5301	0.4902	0.1244	0.1245

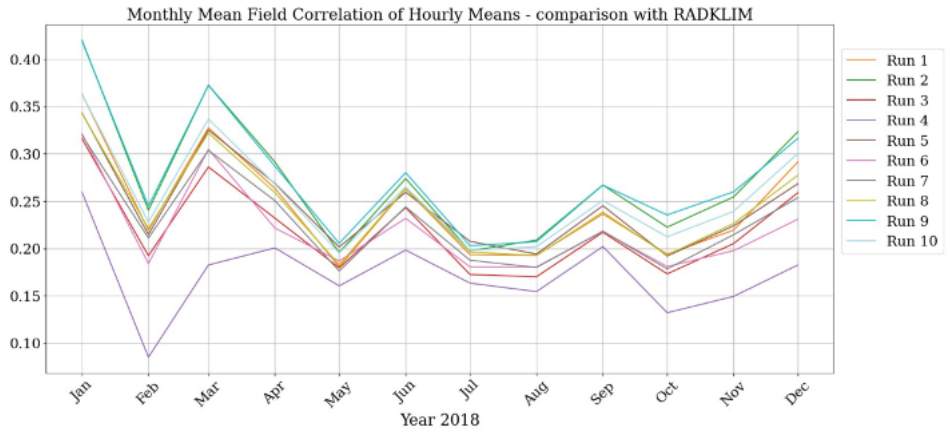


Figure 25: Spatial correlation (using the Pearson correlation coefficient) of the U-Net against the *RADKLIM* dataset over the course of the year 2018. Run 1 (orange line) using just precipitation data is used as the main reference here. Figure taken from [13].

gated daily sparse DDPMs have slightly better values, this comparison is just interesting for contextualization but not fair as it uses 24 data points for each station rather than one.

The U-Net Run 5 can only slightly decrease the baseline U-Net by about 1.1 % whilst the Monte Carlo averaging reduces the sparse DDPM RMSE by 7.5 %. The Monte Carlo run of the aggregated model actually increases RMSE very slightly, but as aggregating over the day is effectively very similar to averaging this result is not that surprising.

Correlation Spatial correlation of the different U-Net runs from [13] are shown in Fig. 25. Correlation coefficient values of the baseline Run 1 range between about 0.37 in January to 0.18 in May, indicating a small positive correlation. The best runs reach a correlation of 0.42 in January.

Comparing this to the DDPM correlation over the course of the year in Fig. 24, it shows that the sparse DDPM model reaches the highest correlation coefficient just over 0.2 in March, the U-Net baseline model from Run 1 exhibits a stronger positive

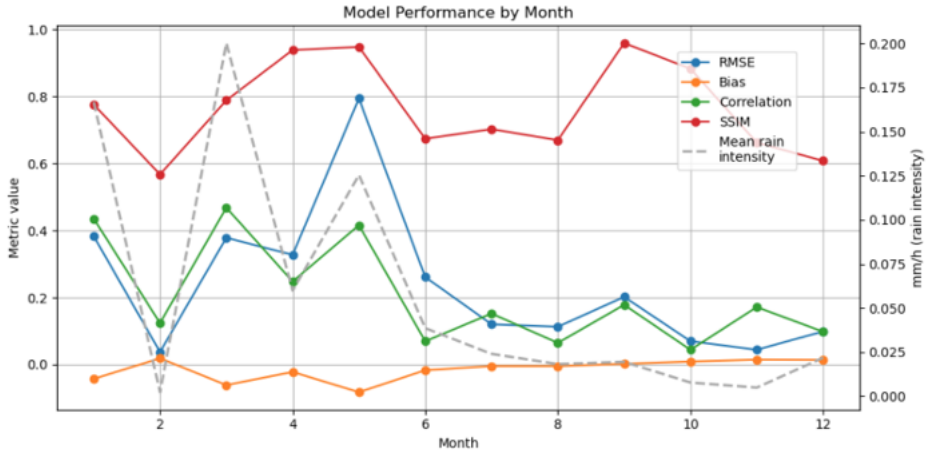


Figure 26: RMSE, Bias, Correlation and SSIM over all months of the year for the Monte Carlo Sparse DDPM version. Again, rain intensity is added on a different scale. The rain intensity peaks differ to the normal version as less time steps were used.

correlation with the ground truth in all months.

Notably, both models have similar correlation peaks over the course of the year, namely in the months January and March, as well as partly slight increases in September and December.

The Monte Carlo version of the Sparse DDPM achieved a correlation coefficient of 0.3289. This significantly increases the normal Sparse DDPM correlation by 59.7 %. The monthly development is shown in Fig. 26.

Correlation appears to be higher in the beginning of the year and lower in the end, and in several months higher than the U-Net model output values.

Total Difference Lastly, total precipitation difference was examined using Equation 28. Hence, a positive value indicates ground truth underestimation whilst a negative value indicates overestimation.

The results are showcased in Table 10. Again, the U-Net and DDPM results are shown with modified versions, for the DDPM the Monte-Carlo version and for the U-Net the best model regarding total precipitation difference, Run 9. This run included precipitation station data of the current, the two previous and the two following time steps as extra input channels.

Also in this metric, that is similar to bias, the U-Net is performing better. However, it underestimates the ground truth whilst the DDPM overestimates it about 3 times as much. Interestingly the Monte Carlo Version is worse here than the DDPM itself, and also conversely underestimates the ground truth.

Table 10: Total precipitation difference of U-Net and DDPM model variants compared to the radar data, given in million mm/h. The DDPM values were proportionally scaled up to $365 \cdot 24 = 8760$ time steps to ensure comparability.

Model type	**U-Net**	**U-Net Run 9**	**DDPM**	**DDPM Monte Carlo**
Total Precipitation Difference (mil. mm/h)	9.8548	6.7933	-31.059	56.9225

5.2.3. HYRAS comparison

After comparing the DDPM to the ground truth in Experiment 1, to REGNIE/HYRAS in Experiment 2 and to the U-Net here in Experiment 3, some final results on the comparison of all three models are presented here.

This is possible due to Filippou et al. also aggregating and comparing their U-Net to the HYRAS dataset (and herby the REGNIE method.

Due to the described partial advantages of the HYRAS dataset over the radar data from the RADKLIM dataset, the HYRAS data is seen as the ground truth just in this section.

RMSE The RMSE results are shown in Table 11. The data shows that the aggregated Monte Carlo version of the sparse DDPM is more similar to HYRAS than the ground truth is in terms of RMSE, whilst this is not the case for the U-Net. Especially in the improved version, the DDPM resembles the HYRAS dataset more than the U-Net does.

Table 11: RMSE comparison of HYRAS against the ground truth, model output, and improved model output in 2018,' given in mm/h. The improved model is the Monte Carlo version for the DDPM and Run 5 for the U-Net (including wind data). HYRAS vs. ground truth has two different values for the models as different time steps were used.

Model	**HYRAS vs Model**	**HYRAS vs Improved model**	**HYRAS vs Ground truth** (Using Model time steps)
DDPM	0.1288	0.1039	0.1212
U-Net	0.1484	0.1474	0.1093

Correlation The Pearson correlation coefficient was also measured against HYRAS in both investigations. The results are given in Table 12. In the U-Net experiment,

the coefficient is only provided on a monthly basis, which is why only a range over the months of the year is given rather than a yearly value.

Table 12: Correlation comparison of HYRAS against model output, improved model output, and ground truth for DDPM and U-Net. Improved model is again the Monte Carlo version for the DDPM whilst for the U-Net Run 9 (including two time steps before and after) had the highest positive correlation with the HYRAS dataset. For the U-Net models, correlations are given as ranges across model time steps as only monthly data was available.

Model	HYRAS vs Model	HYRAS vs Improved model	HYRAS vs Ground truth
DDPM	0.80	0.81	0.88
U-Net	0.18–0.37	0.21–0.43	0.40–0.67

Whilst the DDPM results seem much higher, it is important to note that also the HYRAS vs ground truth correlation is higher, which is only different in both comparisons due to the choice of different time steps.

Evidently, in the time steps chosen for the DDPM, HYRAS was more correlated to the radar ground truth and as both the DDPM and the U-Net were trained with the radar data it might have been easier for the DDPM to achieve high correlation to HYRAS here.

Still, the improved DDPM achieves 92 % of the positive correlation the ground truth yields in relation to HYRAS, whilst the improved U-Net achieves between 0.54 % for the lowest monthly coefficients to 65 % for the highest monthly coefficients, indicating that the DDPM correlates more to HYRAS than the U-Net does.

Total Difference Lastly, total precipitation difference to the HYRAS dataset was examined. Table 13 presents the results. The normal sparse DDPM overestimates precipitation compared to HYRAS, whilst all other machine learning models underestimate HYRAS.

The radar values are however higher than the HYRAS values in line with previous values showing that HYRAS and the U-Net in general rather underestimate precipitation.

Altogether both in the normal and the improved model the Sparse DDPM is slightly closer to the HYRAS dataset in terms of total difference than the U-Net. This is the case even though for the subset of the time steps in 2018 used, the radar data which the model was trained on had a higher total difference (scaled up) than in the rest of the year. Hence, the DDPM appears to be more similar to HYRAS than the U-Net in respect to total precipitation difference.

Table 13: Total precipitation difference between HYRAS and model output, improved model output, and radar, given in million mm/h. Improved models are defined like in the HYRAS correlation comparison (Monte Carlo + Run 9). Values are again proportionally scaled to $24 \cdot 365 = 8760$ time steps for the DDPM models to ensure comparability.

Model	HYRAS vs Model	HYRAS vs Improved model	HYRAS vs RADKLIM
DDPM	-4.8359	1.3055	-2.7306
U-Net	4.9371	2.8747	-0.3648

5.3. Experiment Discussion

For RMSE and Total Difference, the sparse hourly DDPM performs worse than the U-Net when seeing the RADKLIM radar data as ground truth and better when seeing the HYRAS dataset as the ground truth. The correlation is comparable with the radar data and favours the DDPM for the HYRAS dataset comparison.

As RADKLIM was used as the training data for both models and is based on precipitation field data rather than just station data, this shows that the U-Net model performed better than the DDPM.

In the initial comparison to RADKLIM, it was seen that bias was low and structural similarity evident, but RMSE and correlation low. This again indicates that whilst the right kind of precipitation is estimated, the location is often wrong, potentially just by a few kilometres.

This can be observed in the inpainting output in Fig. 23. Precipitation patches are predicted but often in the wrong place. Further, the produced precipitation fields seem structurally less similar to the ground truth than in the first experiment.

This again points out the difficulty of estimating the correct values with such little information. Potentially, the model might improve if learning specific patterns of missing information like done in the U-Net training via masks.

It could be seen with the daily REGNIE model, the correlation coefficient to the radar data was 0.88. As the resulting HYRAS dataset is actually used by weather experts, this can be seen as a target value from which currently, both the U-Net hourly model (best monthly coefficient 0.42) and the DDPM (best monthly coefficient of Monte Carlo version 0.47) differ significantly.

As the main aims of missing value estimation here are to get accurate, location specific values that can inform prediction models, this might make the model outputs unusable.

Measures to improve correlation specifically might yield fruitful here as this specifically focusses on the 'location problem'. Bias is already quite low and RMSE might decrease as correlation increases. This could be done for example by including some term in the loss function that penalizes low correlation rather than just using MSE.

Another option could be to train a binary segmentation DDPM first that does not

learn concrete precipitation values but rather if somewhere there is precipitation or not. Maybe this simpler task could focus the learning on the location rather than the magnitude, and a second model could then estimate magnitude.

The RMSE of the DDPM was three times higher than the U-Net, both for normal and improved versions. In most months, it was five times higher than the average rain intensity.

Whilst RMSE punishes few errors of great magnitude disproportionately, which could be seen in increased RMSE in higher rain intensity periods, the error observed is much higher than the one of the daily HYRAS dataset which serves as a target to beat for the DDPM to actually provide a useful contribution.

Monte Carlo averaging did improve RMSE by 7.5 %. Potentially this could be further increased by averaging over more samples, like 20, 50 or 100. Another option could be to afterwards choose the sample that is most similar (e.g. as measured by MSE) to the average of the samples. The intuition here is that whilst sampling always produces outliers, the average might cancel those out and the sample closest to it will be close to the centre of the actual learned distribution. However, it is unlikely that these steps alone will improve RMSE drastically.

Lastly, the DDPM being more similar to HYRAS than the U-Net points to it being structurally more similar to actual precipitation fields as HYRAS was created by industry experts and had a good SSIM score, however this unfortunately can not be verified directly.

6. Discussion

After presenting and discussing all three experiments, this section summarizes and analyses the results with respect to missing climate data estimation, machine learning and statistical methods for this purpose in general.

The suggested improvements through the experiment discussions are examined in Section 6.1, as well as developing new ones. Section 6.2 presents further ideas if how DDPMs can be used and improved in relation to climate data estimation. Lastly, in Section 6.3 the use of DDPMs and machine learning methods and other methods for the described tasks is reflected upon in general with respect to the results.

6.1. Improvements

This investigation implemented an initial attempt of a DDPM for precipitation data inpainting. Whilst the learned distribution looks realistic in the samples, the DDPM does not perform better than other models.

Throughout the experiment discussions, various improvement ideas were collected and are expanded upon here. The suggestions are clustered to the areas Data, U-Net, Training, Sampling and General Setup and are followed by the authors ranking of likelihood of success.

6.1.1. Data

While data preparation could potentially also be improved upon, e.g. by the use of other scalers or embeddings, the main ideas discussed relating to data here involve the use of more data.

More data can mean longer training, but potentially also less training with more diverse data might be effective.

More Spatio-Temporal Location Data The main difference between the REGNIE method and the machine learning approaches discussed in terms of data is that the former uses spatial coordinates and altitude data (even further processed via the calculation of slope direction and magnitude).

As the U-Net Convolutional layers are applied all over a precipitation field with the same weights and biases, location within the field is not considered (which is also why Patch Diffusion becomes possible, see Section 2.2). Hence, including this as well as altitude in an extra channel with some embedding could be very beneficial.

As a further idea, none of the presented models currently uses temporal information directly, like the point in the day or the year (the REGNIE background field is calculated separately for different months though). As we saw from the rain intensities in 2018, there is temporal patterns throughout the day and the year, that might also be location specific.

Additionally, precipitation patterns might be changing over various years, for example due to climate change. Hence, hour, day of the year (via day and month) and the year itself could be inputted into the U-Net or DDPM as an extra channel through some embedding.

More Meteorological data As discussed, the different runs of the U-Net included various meteorological information like temperature, wind and pressure. Further factors influencing precipitation might be sun radiation or humidity.

While this data is not always available especially when attempting to recreate data going way back, even averages or current values might improve the model performance.

More time steps Perhaps the most intuitive approach is to include station data from previous and future time steps. If there was a lot of rain in Bavaria at one time, and a similarly shaped precipitation pattern in Mecklenburg-Vorpommern four hours later, probably at some point in between, there was a similar pattern across Thuringia.

Filippou et al. propose using this information via added channels, including the two time steps before and after the desired inpainting point in time. This could be done with even more time steps used.

Avoid Importance Sampling The benefits of importance sampling were discussed in Section 2.2. However, it did lead to a lot of information being unused, especially in the hourly model.

We also saw that the model had (in most months) a slight positive bias, so overestimates the precipitation. This might be due to the importance sampling favouring rainy patches in training.

While this could reduce the structural similarity of the datasets, as many zeroes will go into training, it might include more patches with little rainfall and hence improve inpainting for those.

Another option would be to make the importance sampling less strict, so set the boundaries for a patch to be included in the training or validation datasets to be lower.

6.1.2. U-Net

In the implemented thesis, a generic U-Net used for DDPMs by Song et al. [52] was incorporated. In some sense, a DDPM is a U-Net applied in various small increments so should yield at least as good results as a U-Net by itself, which could not be observed here.

This analogy does not work completely as the task that the U-Net is trained on differs between both models, but potentially utilizing some design aspects of the analysed stand-alone U-Net by Filippou et al. as well as other U-Nets used in similar tasks might further improve the model. These ideas are discussed here.

LSTM Incorporation As mentioned, including previous and future time steps might be helpful. Meuer et al. implement this idea in a U-Net not by adding the time steps as extra channels, but by incorporating a long short-term memory (LSTM) layers within the U-Net [36].

These layers were proposed by Shi et al. [50] based on LSTM cells proposed by Hochreiter and Schmidhuber [21] that aim to capture dependencies over time.

Meuer et al. show that this approach is more effective than adding data from 24 time steps in separate channels for the purpose of inpainting missing radar data for precipitation (as opposed to station data).

Partial Convolutions As discussed before, the Partial Convolutions proposed by Liu et al. [32] were used in several similar projects [**?**, 13].

These could be applied in DDPMs in a modified way. As the DDPM learns to predict the noise that was 'added to the data', in training, the fields or patches with missing information could have noise added for the un-missing parts, than partial convolutions could be used to recreate the less noisy field by calculating loss for the valid cells.

No scientific work was found that already implements this but given the success of partial convolutions in U-Nets this seems promising.

Further, this would allow data with partially missing values to be used. In the current U-Net implementation, fields with missing values are ignored in training so this could be a way to increase training data again.

Missing Data Channel Another option to include data with missing values would be to fill the missing values with some placeholder like zero or the mean and then add a mask channel that tells the DDPM which values were missing.

The hope here is that the model learns this relationship, but it might lead do problems if it does not as the fill-value would be overrepresented.

6.1.3. Training

DDPM training is a still evolving field and many implementations have been proposed. In this subsection, two ideas to improve training in the specific DDPM designed for this investigation are discussed.

Different loss functions Evidently, the loss functions of a DDPM is the core of the actual training and backpropagation. Here, we only used MSE. DDPMs are specifically designed to use MSE as the loss function [20] and here the first metric examined was RMSE so having MSE as a loss function is intuitive.

However, the loss function is another big difference between the better performing U-Net and the DDPM so further improvement might be possible here.

Changing the loss function for the DDPM so it still learns the distribution is not as easy as in U-Nets but possible. Lin and Yang describe how they implemented a DDPM with a variant of the perceptual loss used in the U-Net [31].

Use IPS Weights in training In the importance sampling, it was noted that the authors that proposed the scheme did not use loss correction, so multiplying the losses with the inverse acceptance probability, during training as they found "no significant advantage" to this.

However, the model did slightly overestimate values. This could be improved upon by using this weighting also in the training.

6.1.4. Sampling

Another area where the DDPM might perform better is sampling. Three ideas are discussed in the paragraphs below.

Enforcing Constraints During inpainting, DPS is currently used to guide the denoising towards the known stations. Whilst this way the station data is recreated, the resulting rain patches might be otherwise meteorologically improbable.

For example dense rain clouds close to mountains might be more likely as well as chaotic patches aren't the seaside. These are just hypotheses but domain experts

could come up with such rules that could then potentially be incorporated into the sampling process. A similar approach has for example been conducted in a paper on time series sampling [42].

Improved Monte Carlo averaging The Monte Carlo averaging improved the normal model in most cases, for example by 7.5 % in the sparse hourly model. Potentially this could be even taken further. Currently, the averaging is only done over ten time steps (only five for the aggregated model), so as mentioned many more inpaintings could be sampled.

Additionally, the processing of these inpaintings could be improved upon, either by selecting an average of the few inpaintings (or even just one) that is closest to the average.

Another option could be to utilize knowledge of domain experts in which inpainting is chosen, for example by developing methods to identify unrealistic samples or training a model to do so, like a discriminator in generative adversarial networks (GANs).

Synchronize clamping to bias The clamping described in Section 2.2.3 is a powerful tool to limit the range of values sampled. It is restricting the models training but by doing it slowly over the 1000 time steps of inpainting this restriction leaves the model enough freedom.

The values were set using statistical information about the training data. However, they could also be adapted manually to the sampling output itself in order to optimize bias. This way, a bias of near zero could be ensured.

6.1.5. General Setup

Lastly, ideas on the general experiment setups are collected. They cover the general DDPM model training setup.

Cluster areas like REGNIE The REGNIE method described in Section 1.4 calculates different background fields for different spatial clusters within Germany that are meteorologically similar.

This approach has not been implemented in either machine learning method even though it seems plausible that different weather patterns exist in different locations.

Different models could be trained for the different regions. While this has the positive effect that it limits memory usage, the problem of cluster borders would need to be solved somehow

This could e.g. be done with slightly overlapping clusters where the inpainting from one is used as known data in the next inpainting, next to (and potentially overwritten by. the station data of that cluster. The overlapping regions could also be averaged. As this issue has probably been encountered in similar circumstances, a further literature review could be beneficial here.

Notably, this is a general issue as well in the temporal dimension: if there is one inpainting at one time step, and a completely independent one in the next, the resulting time series will be very implausible.

This issue can be fixed by involving time step information either through different channels or other architectural means like the described LSTM layers (see above).

Binary Model Another idea proposed was first training a (non-DDPM) model to decide if there is precipitation at all and then training a separate model to use this output as a separate channel to then fill the precipitation areas with actual values.

This could be helpful in focussing the learning on the exact location of precipitation areas which was a problem in the DDPM, as for example seen in the third experiment.

Test more configurations As this was only an initial exploration, several design decisions were made that could not all be tested individually. For example, the use of patch diffusion and importance sampling could be avoided and the ranges used for layers and model channels could be extended.

Other options like augmentation of patches (mirroring and rotating them) has been experimented with but was not used in the final version due to the risk of this introducing unrealistic patterns into the data, but it might have helped training.

The Optuna search only used 200 trials whilst there would have been infinitely more parameter combinations possible. More beta schedules (like the cosine schedule) could have been tested.

In a deeper exploration all of these options could be explored.

6.1.6. Ranking

To inform future work, the improvements proposed above are ranked by feasibility and likelihood of success based on the research conducted in this study.

1. **More Spatio-Temporal Location Data** As this was the main difference with the better performing REGNIE, and there is a plausible assumption that this data will improve the 'location problem', this measure is most likely to improve results
2. **LSTM Incorporation** LSTM cells were specifically designed to utilise time series and as there is a lot of unused information in the previous and future time steps, this is the most likely way to utilise it for model improvement.
3. **Partial Convolutions** Much of the unused data was due to missing information in the training patches, and this mechanism proved effective for other researchers.

4. **Different loss functions** DDPM training has not yet been specialized architecturally for climate data inpainting which is why improving loss functions could be effective.

5. **Cluster areas like REGNIE** While spatial information could be incorporated via extra coordinate channels, training different models for different regions could be a strong improvement, but the boundary problem needs to be solved.

6. **Improved Monte Carlo averaging**

   Monte Carlo averaging did improve the model, and using more samples or an updated sample selection could help, however it may also make the results more blurry.

7. **More time steps**

   Incorporating the available previous/future time step information should make inpaintings more accurate even though doing it via more channels did not improve results immensely in the U-Net runs.

8. **Test more configurations**

   The hyperparameter search space is huge and there might be a better configuration but as there was already significant optimization in this investigation, this improvement is not prioritized.

9. **More meteorological data**

   Temperature and wind significantly influence precipitation but often the available data will be similarly sparse, and the improvement of using this data again was not significant in the U-Net runs.

10. **Binary Model**

    This interesting idea could be tested with a subset of the data, but as there is little research into this the chance of success is unclear.

11. **Missing Data Channel**

    A missing data channel could help utilize data with missing information but might have unwanted side effects with due to the replacement value.

12. **Synchronize clamping to bias**

    While this will almost certainly perfect bias in the training data, the improvement will be minor and it could be good to leave the model the possibility to predict higher values than the highest recorded ones.

6.2. Ideas

After discussing the concrete improvements for precipitation data estimation with DDPMs, this section discusses further ideas of how they could be used in the field in general and what other aspects of DDPM inpainting were noticed in this investigation. These are discussed in the next paragraphs.

Temporal Downscaling A current challenge with regard to precipitation data is that for various stations, only sub-daily or daily (sometimes even lower) resolution is available, especially when going back multiple decades (see Section 2.1).

While one option to solve this is doing the described inpainting/interpolation of station data with the wanted resolution, this does not use the aggregated information accessible. If in one station, 30 mm were recorded in a day and then the inpainting of all hours of the day does not add up we even know for sure (assuming data collection was done correctly) the estimated values are wrong.

This could be done in a first step via a U-Net that trains a model with one input channel using the daily known values and 24 output channels for the 24 hours of the day. However, it could also be used in a DDPM, where as well 24 input and output channels (potentially also with added channels for spatio-temporal, meteorological or other context information) are used.

In the inpainting, the values of the stations with hourly information could be overwritten, as before. Then, for all specific stations where daily data exists, all estimated values for the spatial cells of theses stations for the 24 hours could be scaled so that the aggregated value matches the measured value. This could also be integrated into the DPS inpainting scheme discussed in Section 2.1.

DDPM with other meteorological data, benchmarks Precipitation data interpolation and inpainting is difficult due to it being localized, non-linear and having a right skewed distribution (see Section 1.1). While DDPM was selected as a model for this specific purpose, it would be good to verify how good DDPMs are here for other meteorological data types.

While there has already been some investigation into this (see [8]), it would be interesting to see how good a DDPM performs for other datasets with a similar station data availability for example for temperature or wind data to verify first if it learns the desired distribution adequately.

This circumstance further points out the need for proper benchmarks to test AI and other model against each other in the field of missing climate data estimation. There are various comparisons of models but they are often applied to very specific problems rather than generalizing the method searching challenge and hence generalizing the results of these investigations for other researchers themselves.

Improved Efficiency Lastly, various of the discussed improvements in Section 6.1, would require more processing power, higher GPU memory and/or longer runtime.

Several of the training and inpainting runs for this investigation lasted several days, sometimes up to a week. The resource use is considerable, and hence it would be desirable to make the model more efficient.

There are several ways with which the current implementation of the DDPM could be made faster. Whilst one option is to sacrifice model complexity, this defeats the exact purpose of the desired improvement. Hence options that simply optimize efficiency without changing the architecture are discussed.

Firstly, smaller data types could be used - the model currently uses float32 tensors, and whilst this accuracy might help tune weights and biases to a more exact magnitude, using float16 or even float8 might speed up the model and reduce memory significantly.

This is a trade-off that could be fairly simply tested and based on this some compromise could be achieved.

Secondly, the model currently only uses parallelization for spreading the batches in training over several GPUs. However, at multiple other spots in the code parallelization might speed up computing.

The most important example here is sampling - the daily aggregate model Monte Carlo run took five days, as the inpainting of a 1024x1024 field used significant memory and resources. The model could have been improved so that the various GPUs be utilized more efficiently.

6.3. Interpretation

Lastly, the main results discussed are highlighted and analysed in a broader sense. In this section, firstly the main scientific contributions of the investigation are summarized. Next, the results are shortly discussed in relation to the field of missing climate data estimation in general. Finally, considerations in going forward with this project are discussed.

6.3.1. Contribution

Throughout this investigation, three main experiments were conducted with various sub-results and discussions. Here, the main scientific contributions are highlighted.

The DDPM deep learning model could neither outperform the statistical REGNIE approach, nor the other deep learning U-Net Model While in some metrics the DDPM results were slightly better, overall the hypothesis, that DDPMs could be better for precipitation data inpainting than the other two models could not be proven. Why might that have been the case? Three possible explanations are suggested:

Firstly, the use of DDPM Inpainting had less time to evolve than the REGNIE methods and U-Nets, the former having already been in use for "many years" in

2013 [45], the latter having been invented five years before the U-Net. This was only an initial investigation with limited parameter optimization and basic architecture. Future investigations could achieve better results.

Secondly, in terms of the REGNIE comparison, the DWD method used other data sources like location and altitude data (and in turn did not use radar data directly) making both models less comparable.

Thirdly, concerning the U-Net comparison, the worse performance might have been due to a trade off between complexity and performance. U-Nets are less complex than DDPMs and can learn faster. Hence, the U-Net used had more layers than the DDPM and used more data during training.

Whilst the exact runtime and computing power used for the U-Net training is unknown, the DDPM layers and channels were limited by runtime and resources. The results of this investigation suggest that the added architectural benefit and actual denoising of DDPMs did not cancel out the effects of a more simple U-Net used as a model in DDPM and less training data used.

The DDPM Monte Carlo performed well and in some situations comparable to the other models, showcasing the potential for DDPMs to surpass the other models. The Monte Carlo Version of the DDPM had a better bias score than the HYRAS dataset in comparison with the ground truth radar data and comparable other metrics, including an RMSE that was only 1.2 % over the same HYRAS error. In the sparse hourly model, its correlation was comparable but the other metrics were worse to the U-Net.

This showcases on the one hand that the output of probabilistic models themselves sometimes can be outliers, on the other hand it shows the advantage in estimating a distribution that can then outperform deterministic models.

As discussed there are still various options on how exactly this variant is implemented and it might be possible to surpass REGNIE and the U-Net in the future.

In the daily model, the Monte-Carlo DDPMs results were comparable to those of HYRAS whilst in the sparse hourly model they were, in most categories, clearly less good than the U-Net scores. As in the hourly model only 2% of the data was used, this indicates that the DDPM architecturally is not worse than the U-Net and close to achieving better scores than REGNIE. Hence, with the proposed improvements, the DDPM might become the best performing model.

The DDPM output was closer to HYRAS than the U-Net output. The HYRAS dataset was developed with the REGNIE method by meteorological experts over years for the purpose of precipitation data estimated. The DDPM and U-Net were both trained on radar data, and the DDPM achieved more similarity to HYRAS than the U-Net did.

This might be due to a higher structural similarity with the precipitation data due to the diffusion learning process. As there is no information on this available this could only be investigated through a reimplementation of it.

It is important to note that this was not the described task of either model to learn and hence not the main focus of the investigation. Yet, it highlights the importance to verify results with a variety of metrics.

The DDPM using hourly aggregate data beats the current daily HYRAS model. The aggregate hourly DDPM model did beat the HYRAS model in all metrics except structural similarity when comparing the datasets with the ground truth in the second experiment (Section 4). A Monte Carlo version might have even improved the results further. Hence, ***the DDPM performed better than HYRAS here***.

For the sparse hourly model in experiment three, this was no longer true. It attempted to recreate a situation of sparse (hourly) station coverage, like it was in the 1950s, for the inpainting setup. Unlike hourly stations, the number of daily stations did not vary as much as shown in Fig. 4 in Section 2.1.

Hence, for those years the HYRAS dataset is still better than the DDPM Aggregate model. Further, as discussed, the comparison is unfair as the REGNIE method producing the HYRAS dataset only used daily, not hourly data.

However, this shows the potential of using such data - the REGNIE method could be applied to the hourly station data and then aggregated as well to improve the missing value estimation. Presumably, this would achieve better results than the DDPM due to this being the case for a fairer comparison at the daily resolution (not taking into account that REGNIE had the advantage of REGNIE using location and altitude data).

6.3.2. Field Context

In this section, reflections on the investigation in general are discussed relating to the use of machine learning techniques in climate science in general as well as the comparison of such techniques.

Domain specific knowledge matters. The REGNIE method achieved better results than the two machine learning methods, when assessing its resemblance to the radar data, even though the latter were specifically trained on this task.

While this might have various technical reasons, it is important to point out that the process of creating the models were different. The REGNIE method was developed and verified over many years by domain experts, while the machine learning approaches tried to directly apply a modern technique to the problem discussed.

The REGNIE method differs in the data sources being utilized and the approach in general - making different background fields for different months of the year and spatial clusters in Germany. These ideas can be reimplemented by computer scientists, nevertheless it is important to point out that out of a mere computer science perspective solutions might not become easily apparent.

Machine learning methods did not outperform the statistical approach. Next to the importance of domain experts, it is also interesting to point out that a statistical method outperformed modern machine learning models. This is not an uncommon phenomenon.

As Trevor Hastie pointed out in 2009: "It seems that whatever exotic tools are the rage of the day, we should always have available these two simple tools" [17]. While he was talking about linear and quadratic discriminant analysis, the underlying message that not all new AI tools are immediately better and often overestimated, is still relevant today.

Further, while REGNIE might in some cases create more blurry precipitation fields than the ML methods, it is clear what the underlying assumptions are while the AI results are harder to interpret as they are a blackbox.

Yet, the results also showed the potential of improvement for the machine learning models, and in other research they did perform the desired tasks well (see [**?**]) so it will be interesting to see the field develop in the coming years.

6.3.3. Next steps

Lastly, it is reflected what considerations need to be made for using DDPMs and machine learning techniques in future work.

Balancing complexity and resource use Retrospectively it is hard to quantify exactly how much energy was used for this investigation. Both final models ran several days over multiple GPUs, as well as every inpainting experiment and the Optuna hyperparameter optimization.

This is only considering the runs that were used for the final results section, there were various more test and debugging runs necessary as well as discarded runs due to mistakes in the implementation.

Various of the improvements suggestions covered in Section 6.1 would increase resource use, and while efficiency measures were proposed in Section 6.2 future work on DDPMs in climate data estimation will still require much more energy than statistical methods.

This poses the question whether the research is worth it. It is only answerable on a case by case basis but these considerations should be taken into account when attempting a new investigation, especially given that the methods proposed here were not immediately successful.

While the DDPM and machine learning results in general were mixed, there is loads of potential. Further research is likely fruitful. The field of machine learning in climate science is only evolving. Whilst the statistical method was still the best in this investigation, the machine learning approaches had comparable and partly better results in various categories.

The DDPM had similar correlation to the U-Net, resembled the in-use HYRAS more and the Monte Carlo method outperformed the REGNIE method in bias and had very similar scores in most other metrics. Especially with the use of coordinate and altitude data, that REGNIE already uses, a better performance is likely.

7. Conclusion

This investigation examined the use of artificial intelligence techniques in missing climate data information. It compared a statistical approach, the REGNIE method currently used by the DWD, with machine learning inpainting methods based on a U-Net and a Denoising Diffusion Probabilistic Model (DDPM).

The results showed that in this initial attempt, the statistical method achieves better results compared with the ground truth radar data. The U-Net performed better than the DDPM. An improved version of the DDPM, the Monte-Carlo version, achieved comparable results with the U-Net.

Several ideas and improvements for the DDPM implementation were suggested. Given that the use of DDPMs for climate data inpainting is still evolving and that the development of the REGNIE has been perfected over the last decades and will probably not improve much more, this points out the immense potential in this field.

Concretely, the use of more spatio-temporal location data, the incorporation of station data from more time steps via LSTM integration and the use of partial convolutions was recommended in the discussion.

Further, general ideas for the use of AI methods in climate data estimation, like the introduction of benchmarks, the involvement of domain experts and the consideration of resource use was suggested.

A. Code

The code developed for this thesis is available at: https://github.com/Merjo/DDPM_Inpainting

B. U-Net Model Investigation

The thesis by Danai Filippou describing the implementation of the U-Net that led to the conference poster by Filippou et al. (see [13]) can be found here: https://github.com/Merjo/DDPM_Inpainting/blob/main/docs/Master_Thesis_Danai_Filippou.pdf.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[2] J. Behrendt and K. Zimmermann. Qualitätskontrolle historischer klimadaten. Klimastatusbericht, Deutscher Wetterdienst (DWD), 2008.

[3] Marcelo Bertalmío, Guillermo Sapiro, Vicent Caselles, and C. Ballester. Image inpainting. pages 417–424, 01 2000.

[4] Nils Bochow, Anna Poltronieri, Martin Rypdal, and Niklas Boers. Reconstructing historical climate fields with deep learning. *Science advances*, 11:eadp0558, 04 2025.

[5] Peter Bonate. A brief introduction to monte carlo simulation. *Clinical pharmacokinetics*, 40:15–22, 02 2001.

[6] G. Box and David Cox. An analysis of transformations (with discussion). *Journal of the Royal Statistical Society, Series B*, 26:211–252, 01 1964.

[7] Bundeszentrale für politische Bildung Redaktion. Nach der flut an der ahr 2021. Hintergrund aktuell, bpb.de, July 2021. Accessed: 2025-07-17.

[8] Christian Burmester, Philipp Hess, and Niklas. Boers. Reconstructing historical temperature fields using diffusion-based generative machine learning. In *Proceedings of the 13th German Climate Conference (Deutsche Klimatagung)*, 2024. Contribution ID: DKT-13-60.

[9] Andrew Charbonneau, Katherine Deck, and Tapio Schneider. A physics-constrained neural differential equation framework for data-driven snowpack simulation, 2025.

[10] Hyungjin Chung, Jeongsol Kim, Michael T. Mccann, Marc L. Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems, 2024.

[11] Deutscher Wetterdienst. Climate data for direct download. Deutscher Wetterdienst (DWD), Climate Data Center (CDC), 2024. Accessed: 2026-01-05.

[12] Deutscher Wetterdienst. Tägliche niederschlagsbeobachtungen für deutschland, 2024. Daily precipitation observations from DWD stations in Germany. Licensed under CC BY 4.0.

[13] Danai Filippou, Étienne Plésiat, Johannes Meuer, Hannes Thiemann, Thomas Ludwig, and Christopher Kadow. Machine learning-driven infilling of precipitation recordings over germany. In *EGU General Assembly 2023, Vienna, Austria*, pages EGU23–9191, 2023. Based on a master's thesis by Danai Filippou that is linked in the appendix.

[14] Thomas Forbriger. Lecture notes on "physics of seismic instruments": Power spectral density and rms-amplitude. Lecture notes, Karlsruhe Institute of Technology (KIT), 2020. Accessed: 2026-01-13.

[15] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style, 2015.

[16] Nikolaos Giakoumoglou, Eleftheria Maria Pechlivani, and Dimitrios Tzovaras. Generate-paste-blend-detect: Synthetic dataset for object detection in the agriculture domain. *Smart Agricultural Technology*, 5:100258, 2023.

[17] Trevor Hastie. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 01 2009.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[19] Philipp Hess, Michael Aich, Baoxiang Pan, and Niklas Boers. Fast, scale-adaptive and uncertainty-aware downscaling of earth system model fields with generative machine learning. *Nature Machine Intelligence*, 7:363–373, 03 2025.

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 06 2020. Preprint.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 11 1997.

[22] Svenja Hoffmann. Niederschlagsdisaggregation stand 07.02.2025. Presented at the internal BMDV Expertennetzwerk Jour Fixe, Deutscher Wetterdienst, February 10, 2025. Figure reproduced with permission., 2025.

[23] I. Holleman, D. B. Michelson, G. Galli, U. Germann, and M. Peura. Quality information for radars and radar data: Deliverable: Opera 2005 19. Opera work package 1.2, OPERA, 2006.

[24] Julong Huang, Chuhan lu, Dingan Huang, Yujing Qin, Fei Xin, and Hao Sheng. A spatial interpolation method for meteorological data based on a hybrid kriging and machine learning approach. *International Journal of Climatology*, 44:5371–5380, 10 2024.

[25] Aniketh Iyengar, Jiaqi Han, Boris Ruf, Vincent Grari, Marcin Detyniecki, and Stefano Ermon. Energy scaling laws for diffusion models: Quantifying compute and carbon emissions in image generation, 11 2025. Preprint.

[26] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155, 2016.

[27] Christopher Kadow, David Hall, and Uwe Ulbrich. Artificial intelligence reconstructs missing climate information. *Nature Geoscience*, 13, 06 2020.

[28] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models, 06 2022. Preprint.

[29] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[30] Jin Li and Andrew Heap. Spatial interpolation methods applied in the environmental sciences: A review. *Environmental Modelling Software*, 53:173–189, 03 2014.

[31] Shanchuan Lin and Xiao Yang. Diffusion model with perceptual loss, 2025.

[32] Guilin Liu, Fitsum A. Reda, Kevin J. Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. *CoRR*, abs/1804.07723, 2018.

[33] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[34] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Gool. Repaint: Inpainting using denoising diffusion probabilistic models, 01 2022. Preprint.

[35] Roman Maier, Gerald Krebs, Markus Pichler, Dirk Muschalla, and Günter Gruber. Spatial rainfall variability in urban environments—high-density precipitation measurements on a city-scale. *Water*, 12(4), 2020.

[36] Johannes Meuer, Laurens Bouwer, Frank Kaspar, Roman Lehmann, Wolfgang Karl, Thomas Ludwig, and Christopher Kadow. Infilling of missing rainfall radar data with a memory-assisted deep learning approach, 07 2024.

[37] Ana Militino, M. Ugarte, and Unai Pérez-Goya. *Machine Learning Procedures for Daily Interpolation of Rainfall in Navarre (Spain)*, pages 399–413. 01 2023.

[38] Hamidreza Mosaffa, Luca Ciabatta, Paolo Filippucci, Mojtaba Sadeghi, and Luca Brocca. Hr-precipnet: A machine learning framework for 1-km high-resolution satellite precipitation estimation. *Journal of Hydrology*, 658:133217, 03 2025.

[39] Christoph Müller, Manuela Nied, Matthias Voigt, Christian Iber, Tilmann Sauer, Thomas Junghänel, Andreas Hoy, and Heike Hübener. Starkniederschläge – entwicklungen in vergangenheit und zukunft. Technical report, Kooperation KLIWA, Germany, 2019.

[40] Congyi Nai, Baoxiang Pan, Xi Chen, Qiuhong Tang, Qingyun Duan, Bo Lu, Ziniu Xiao, and Xingcai Liu. Reliable precipitation nowcasting using probabilistic diffusion models. *Environmental Research Letters*, 19:034039, 02 2024.

[41] A. K. Nain. Generating images of flowers with denoising diffusion probabilistic models. `https://keras.io/examples/generative/ddpm/`, November 2022. Accessed: 2026-01-12.

[42] Sai Shankar Narasimhan, Shubhankar Agarwal, Litu Rout, Sanjay Shakkottai, and Sandeep P. Chinchali. Constrained posterior sampling: Time series generation with hard constraints, 2025.

[43] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models, 2021.

[44] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703, 2019.

[45] Monika Rauthe, Heiko Steiner, Ulf Riediger, Alex Mazurkiewicz, and Annegret Gratzki. A central european precipitation climatology – part i: Generation and validation of a high-resolution gridded daily data set (hyras). *Meteorologische Zeitschrift*, 22:235–256, 07 2013.

[46] Suman Ravuri, Karel Lenc, Matthew Willson, Dmitry Kangin, Remi Lam, Piotr Mirowski, Megan Fitzsimons, Maria Athanassiadou, Sheleem Kashem, Sam Madge, Rachel Prudden, Amol Mandhane, Aidan Clark, Andrew Brock, Karen Simonyan, Raia Hadsell, Niall Robinson, Ellen Clancy, Alberto Arribas, and Shakir Mohamed. Skilful precipitation nowcasting using deep generative models of radar. *Nature*, 597:672–677, 09 2021.

[47] Ricardo Reinoso-Rondinel, Martin Rempel, Markus Schultze, and Silke Troemel. Nationwide radar-based precipitation nowcasting - a localization filtering approach and its application for germany. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 15:1670–1691, 01 2022.

[48] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[49] Jörn Seidel, Merle Göddertz, Han Lay, Axel Klauwer, Raimund Groß, Jeanette Erfurth, and Julia Linn. Chronik – ahrtal hochwasser-katastrophe. WDR Online Reportage, September 2021. First published: 2021-09-10; last updated: 2024-07-12. Accessed: 2025-07-17.

[50] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai kin Wong, and Wang chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting, 2015.

[51] Zhenru Shu, Mike Jesson, and Mark Sterling. Nonlinear dynamic analysis of daily rainfall variability across the uk from 1989 to 2018. *Journal of Hydrology*, 603:126849, 2021.

[52] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *CoRR*, abs/2011.13456, 2020.

[53] Jörg Steinert, Patrick Tracksdorf, and Dirk Heizenreder. Hymec: Surface precipitation type estimation at the german weather service. *Weather and Forecasting*, 36:1611–1627, 08 2021.

[54] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions, 09 2021.

[55] Robin Vinod. Attention gates in u-nets: A detailed explanation of the attention u-net, May 01 2020. Medium article, accessed January 11, 2026.

[56] Zhendong Wang, Yifan Jiang, Huangjie Zheng, Peihao Wang, Pengcheng He, Zhangyang Wang, Weizhu Chen, and Mingyuan Zhou. Patch diffusion: Faster and more data-efficient training of diffusion models, 04 2023. Preprint.

[57] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[58] Ruifang Wei and Yukun Wu. Image inpainting via context discriminator and u-net. *Mathematical Problems in Engineering*, 2022:1–12, 05 2022.

[59] P. S. Wilson and R. Toumi. A fundamental probability distribution for heavy rainfall. *Geophysical Research Letters*, 32(14), 2005.

[60] Tanja Winterrath, Christoph Brendel, Mario Hafer, Thomas Junghänel, Anna Klameth, Ewelina Walawender, Elmar Weigl, and Andreas Becker. *Erstellung einer radargestützten Niederschlagsklimatologie*. Number 251 in Berichte des Deutschen Wetterdienstes. Deutscher Wetterdienst, 2017. `http://nbn-resolving.de/urn:nbn:de:101:1-20170908911`.

[61] Carissa Wong. How ai is improving climate forecasts. *Nature*, 628(8009):710–712, April 2024.

[62] Yuxin Wu and Kaiming He. Group normalization, 2018.

[63] Guanhua Zhang, Jiabao Ji, Yang Zhang, Mo Yu, Tommi Jaakkola, and Shiyu Chang. Towards coherent image inpainting using denoising diffusion implicit models. 04 2023. Preprint.