Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Bachelor Thesis

Title of the Thesis // Titel der Arbeit

Scientific software development of a Convolutional Neural Network for the reconstruction of missing data from a weather measurement station using numerical model data.

Wissenschaftliche Softwareentwicklung eines Convolutional Neuronal Networks für die Rekonstruktion fehlender Daten einer Wettermessstation unter Verwendung von Numerischen Modelldaten

---

Academic Degree // Akademischer Grad
Bachelor of Science (B.Sc.)

Author's Name, Place of Birth // Name der Autorin/des Autors, Geburtsort
Timo Wacke, Hamburg

Field of Study // Studiengang
Computing in Science (Physics Specialization)

Department // Fachbereich
Computer Science // Informatik

First Examiner // Erstprüferin/Erstprüfer
Prof. Dr. Thomas Ludwig

Second Examiner // Zweitprüferin/Zweitprüfer
Dr. Christopher Kadow

Matriculation Number // Matrikelnummer
7434883

Date of Submission // Abgabedatum
10.06.2024

Abstract

This bachelor thesis presents a method to reconstruct missing temperature measurements from low-cost weather stations using ERA5 Reanalysis data and a Convolutional Neural Network (CNN). The work aims to address challenges posed by sparse weather station coverage by supporting low-cost stations with gap-filling. The proposed method involves training a CNN model to estimate the weather stations hourly measurements based on the surrounding 8x8 grid data from the ERA5 Reanalysis. The trained model can then reconstruct missing station data, outperforming traditional numerical methods in computational efficiency. The thesis describes the conceptual framework, including data acquisition, preprocessing, CNN training, model validation using metrics like RMSE and correlation coefficient, and application for filling missing data. It provides background on CNNs and atmospheric reanalysis. The results evaluate the models' performance across stations in different environments. The implementation covers the pipeline for training models, reconstructing data, data handling, interfacing with the Copernicus API, and preprocessing steps. It also discusses process orchestration with a web interface and API for training, infilling, and validation.

Zusammenfassung

Acknowledgements

I am deeply grateful for the opportunity to be a part of the DKRZ Data Analysis team and I'd like to express my gratitude to the following people who have supported me throughout this project. I want to express my heartfelt thanks to the entire team for warmly welcoming me and providing unwavering support throughout my time here. I'll cherish the memories of our Friday meetups and the fun games we played together. I would like to extend special thanks to Danai for her invaluable support and guidance, and to Etor for his consistent backing throughout the project. Additionally, Max deserves recognition for his insightful feedback and engaging discussions. I am also thankful to Mo for his helpful feedback and energizing morning runs. I'm also immensely thankful to Chris for his exceptional mentorship and continuous support. His guidance, discussions, and encouragement have significantly influenced the direction of my work. I appreciate the 3D-PAWS team for granting access to weather station data and providing valuable feedback. My gratitude also goes to Alisa and Sebastian for their consistent support and constructive feedback. A special mention to Toli for ensuring I had the necessary time to dedicate to this project. Lastly, I want to thank Franko for making long nights at the office fun.

Contents

List of Figures

1 Introduction

Weather station density varies greatly across the globe, depending on population density, economic development, and access to nearby infrastructure [1]. While any weather station can experience downtime, the reliability of weather stations in regions with low station density is often low as well [2]. So not only is downtime in regions where data is limited more likely, but it is also more impactful because there are fewer neighboring stations to help compensate for the missing data.
A denser, more reliable network would benefit weather forecasting, helping to evacuate populations timely before natural disasters [3] and would contribute globally available climate data. An innovative approach to increase the density of weather stations could be to use low-cost weather stations that could be 3D-printed and assembled by the local population [3]. This is the approach taken by the 3D-PAWS project (see section 2). Either way, low-cost weather stations have reliability issues and are prone to downtime. In this work, out of the many globally deployed stations of the 3D-PAWS project, three stations that have many years of history have been selected for the evaluation of the proposed method. One station is located on an island surrounded by ocean, Barbados, one in a city, Vienna, and one in a mountainous region, Boulder, Colorado. In Figure 1.1, temperature measurements of the Vienna station since its deployment in 2017 are shown, illustrating an extreme case of downtime.

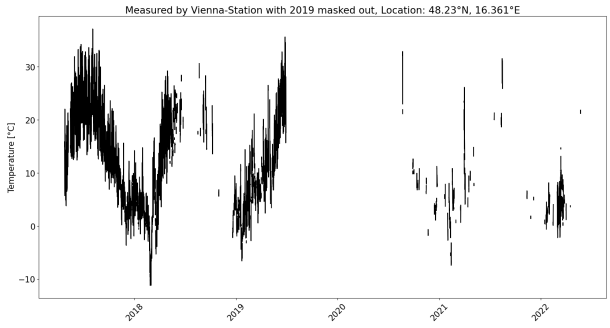


Figure 1.1: Temperature measurements of a 3D-printed weather station in Vienna, Austria.

In light of the challenges posed by sparse weather station coverage, novel methodologies are required to address the reconstruction of missing weather data. Machine learning offers a promising alternative to traditional numerical reconstruction methods, such as kriging, that rely on neighboring station data and are often computationally intensive [4]. Numerical alternatives like Weather Research and Forecasting (WRF) models provide greater accuracy than kriging but are even more computationally demanding [5]. The application of machine learning, in this case, would be to relate coarse-resolution numerical

reanalysis data, which in itself can not accurately describe the local characteristics, to the local patterns at the weather station. This would allow for independent operation without relying on neighboring stations or additional data sources. It can directly utilize the global reanalysis data as input, outperforming numerical methods in terms of computational resource requirements by orders of magnitude [6, 7, 8] as the application of the trained machine-learning model. By leveraging available local data, these techniques, such as Convolutional Neural Networks (CNNs), can be trained to estimate weather conditions at a designated time by assimilating global numerical weather model data. Despite the inherent blurriness of aerial data provided in grid cells, these models are anticipated to discern and adapt to local weather patterns such that they become capable of transferring knowledge from the meta situation to the local situation. This paper aims to achieve reconstruction using that approach, which will be further explained in section 3.
The reanalysis of choice in this endeavor is the ECMWF Reanalysis v5 (ERA5), which covers the globe in grid cells of 0.25° x 0.25°. The data is available in hourly timesteps from 1940 to the present and contains a wide range of variables, such as temperature, precipitation, wind speed [9].
To prove the concept, it is likely most straightforward to start with temperature data, meaning the 2-meter temperature variable from the ERA5 reanalysis will be used as input to the neural net.

2 3D-printed Weather Stations

3D-printed Automated Weather Stations (3D-PAWS) represent an innovative approach to enhancing meteorological observation networks, particularly in regions with limited data availability. Utilizing 3D printing technology, these stations are constructed from lightweight and durable plastic, facilitating easy deployment and maintenance. Equipped with sensors to measure parameters such as temperature, wind, solar radiation, and precipitation, 3D-PAWS ensure comprehensive weather data collection [10].
At the heart of the 3D-PAWS system lies a Raspberry Pi, responsible for sensor control and data logging. Powered by solar panels and batteries, these stations can operate autonomously in remote areas. Data transmission is facilitated through either cell or satellite modems, enabling real-time updates accessible under open access policies. This approach not only ensures continuous data flow but also promotes transparency and accessibility in meteorological data sharing [10].
Cost-effectiveness is a significant advantage of 3D-PAWS. Structural components for

these stations are locally sourced, with an approximate cost of $100. The Raspberry Pi, together with all needed micro-sensors, are off-the-shelf products and cost below $200. Communication costs remain minimal, with cell modems priced at $30 [10].

Recent advancements in 3D printing have substantially reduced the production costs of these weather stations. With capable 3D printers now available for around $500, the production and deployment of 3D-PAWS have become more accessible and affordable. This democratization of weather station technology holds the potential to revolutionize meteorological observation, especially in regions where traditional stations are impractical or prohibitively expensive.

In summary, 3D-PAWS offers a significant advancement in meteorological observation, combining affordability, ease of deployment, and comprehensive data collection capabilities. By leveraging 3D printing and modern sensor technology, these stations can address critical gaps in weather data networks worldwide, thereby improving weather prediction and climate research efforts [3].

The primary objective of the 3D-PAWS initiative is to save lives through improved early warning systems for catastrophic events. The project is sponsored by the University Corporation for Atmospheric Research (UCAR) and the US National Weather Service International Activities Office (NWS IAO), with support from the USAID Office of U.S. Foreign Disaster Assistance (OFDA) [11].

3D-PAWS systems are being rigorously evaluated at different locations, including a testbed field site in Colorado, where one of the weather stations included in this work is located. These evaluations assess sensor performance in diverse climatic conditions, ensuring the reliability and accuracy of data collected by the stations. [3]

Pilot networks of 3D-PAWS have been deployed in the United States and over ten other countries globally. Real-time data from these stations are accessible through the CHORDS project data servers, facilitating their utilization in hydrometeorological applications worldwide [11].

The observations provided by 3D-PAWS have diverse applications, including regional weather forecasting, early alert systems for natural disasters, agricultural monitoring, and health surveillance. These applications underscore the potential impact of 3D-PAWS in mitigating risks associated with weather-related events and enhancing resilience in vulnerable communities. [11].

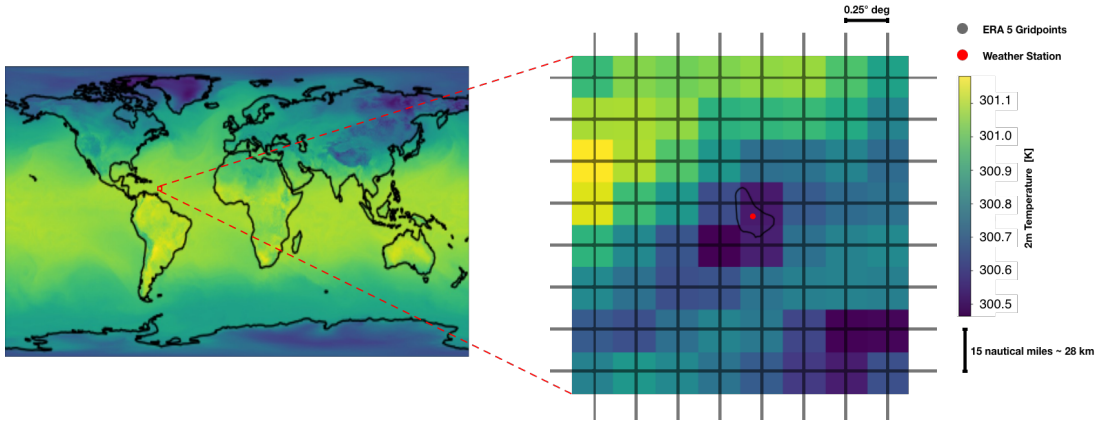


Figure 3.1: 8x8 grid-points of ERA5 with the 2-meter temperature for 2020-06-23 19:00 UTC in the area of a weather station on Barbados

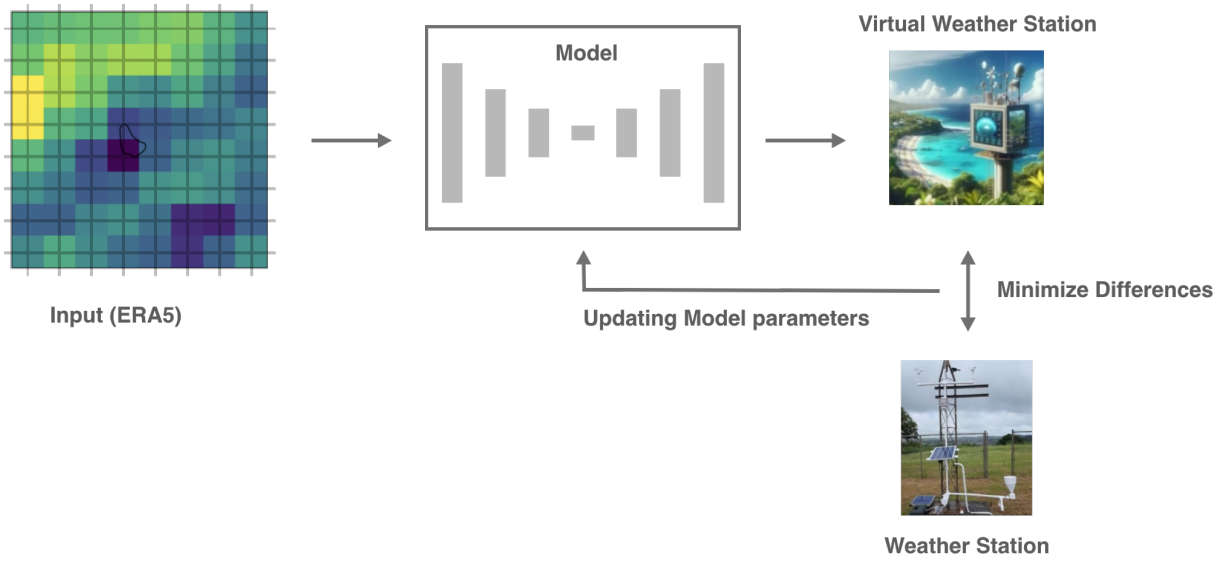


Figure 3.2: Conceptual framework using supervised learning.

3 Conceptual Framework

3.1 Concept

The conceptual framework of the proposed method is illustrated in Figure 3.2. Applying the local patterns at the weather station location on top of ERA5 would reconstruct the temperature data at the station. The idea is that a Convolutional Neural Network would learn to recognize the relevant patterns in a regional cutout of ERA5 data in the station area. Based on the regional ERA5 data, the network would then predict the temperature at the weather station. To illustrate with an example in what form these discrepancies can emerge: In the case of Barbados, the grid cells of the ERA5 data all lay primarily in

the ocean, meaning the diurnal cycle has a much lower amplitude than at the weather station that is located on land. The neural network would need to learn how to detect the diurnal cycle phase based on the 64 grid points and then adjust the temperature values accordingly. Besides this obvious difference between ERA5 and the measurements at Barbados, there are most likely many more dynamic local effects to be learned and adjusted for.
Since the ERA5 data is available globally and hourly, any weather station's missing hour could be reconstructed simply with a model that has been trained for the respective station. A supervised learning approach will be used to train the model (see Figure 3.2). Therefore, the model's weight will be adjusted during training so that the difference between the prediction based on an ERA5 input and the actual measurement is minimized. The design decision to make the model based on one timestep of input for one timestep of output and not on a sequence of timesteps has the following advantages: It allows for simplicity in training, as no sequence handling is needed but simply any single timestep can be used for training. It also allows for flexibility in application, as the model can be applied to any single timestep of ERA5 data to reconstruct the temperature at the weather station at that time. The disadvantage is that the model can not learn from the sequence of timesteps, meaning it can not learn from the temporal evolution of the weather patterns. However, that is not anticipated to be a problem since all that information is already included in the ERA5 data. Therefore, when reconstructing missing data, the model is applied to the ERA5 data hour by hour, and the result is a series of hourly predictions that are not connected in time.

3.2 Data Acquisition and Preprocessing

Upon obtaining a dataset from a weather station, it is determined where temperature data is missing. While the weather station dataset is minute-based, data could be missing only for a few minutes within an hour instead of the full hour. This would raise the question of how many missing values are acceptable to not mark the hour as missing. Sure is, that if all temperature values are missing during an hour, the hour is marked as missing. The ERA5 data then needs to be cropped to the neighboring 8x8 grid cells, while centering the cutout as close to the weather station as possible. The available longitudes and latitudes in the ERA5 model are spaced by 0.25° along the latitude and longitude from each other, when 8x8 grid cells are selected it needs to be assured that the grid points are such selected that the coordinates of the weather station are between the 4th and 5th grid point in each dimension. After cropping the ERA5 dataset geographically, the data needs to be cut and

divided along the time axis to match the weather station data, leading to two datasets: one with all the hours marked as missing and one with all the hours marked as present. Until the model is trained, only the dataset with all the hours marked as present will be used.
As a result, we have a dataset pair of weather station measurements and ERA5 data, that are coherent in time and space.

3.3 Training of the Convolutional Neural Network

To determine after the training if and to which extent the model learned to reconstruct the missing data, the dataset pair is split again along the time axis into a pair of station with ERA5 data for training and a pair of station with ERA5 data for validation. Thus we can later let the model reconstruct values that have actually been measured but haven't been included in the training so we then can validate how successful the reconstruction is. With the datasets prepared, the next phase involves configuring and training the Convolutional Neural Network (CNN) for the temperature reconstruction task. The CNN architecture is tailored to accept input in the form of 8x8 grid cells centered around the weather station's location. Employing a supervised learning approach, the CNN is trained using pairs of hourly temperature data from the weather station and corresponding grid cell data from ERA5. The training process iteratively feeds batches of data into the CNN, fine-tuning its parameters to minimize prediction errors and optimize accuracy in reconstructing missing temperature values.

3.4 Validation of the trained model

Following the training phase, the CNN's performance is evaluated using the validation set. The model's capacity to accurately reconstruct missing temperature data at the weather station is scrutinized against ground truth values. This evaluation step serves to gauge the CNN's proficiency in capturing intricate weather patterns and producing precise temperature estimations. For that, the root mean squared error (RMSE) and the correlation coefficient are calculated. The RMSE is a measure of the differences between predicted and observed values, while the correlation coefficient quantifies the strength and direction of the linear relationship between the two datasets. An obvious choice as a time range for the evaluation would be to cut out one complete year so that the model can be evaluated over the full range of seasons and weather conditions.

3.5 Application for infilling

Upon successful training and validation, the model is trained again with the measurements that have been excluded before for the benefit of validation. After training on the complete data, the CNN can be used to fill gaps. Fundamentally, the predictions can be made for any list of timesteps. Firstly, the needed ERA5 data, for the respective timesteps will be obtained, and secondly cropped in the same way to the geographical region as during training and then given as input to the model. The result is a list of temperature values that are not connected in time but are the model's prediction for the temperature at the weather station at the respective time. To predict the temperature at all those times where measurements are missing, preprocessing work can be shortened, with the available datasets already prepared in 3.2. For the training the ERA5 dataset was already split into one with all hours marked as missing and one with all hours marked as present. Thus, the model can directly be applied to the dataset with all hours marked as missing and then directly filled into the original measurements dataset.

4 Theoretical Background

4.1 Convolutional Neural Networks

CNNs are a type of deep learning architecture inspired by the visual cortex of animals. [12, 13]. They are designed to efficiently capture spatial and temporal dependencies in data through the use of learnable filters and hierarchical feature representations. [12]. Through the use of convolutional layers instead of fully connected layers, the architecture is able to preserve the spatial structure of the input data, making it particularly well-suited for image and video data. This approach not only simplifies pattern detection but also implies a reduction in the number of parameters, which minimizes the necessary computational resources.

Application of CNNs in climate science has yielded several notable contributions [14, 15], including the reconstruction of the El Nino event of 1877 by Kadow et al. despite extremely limited data availability. [16]

In the context of this work, an architecture introduced by ([17]) is used, which consists of an encoder and a decoder part as seen in Figure 4.3c. Due to its shape, it is called U-Net. The encoder part consists of convolutional layers and pooling layers, while the decoder part consists of upscaling layers and, in this case, skip connections as proposed by [18].

Convolutional Layer

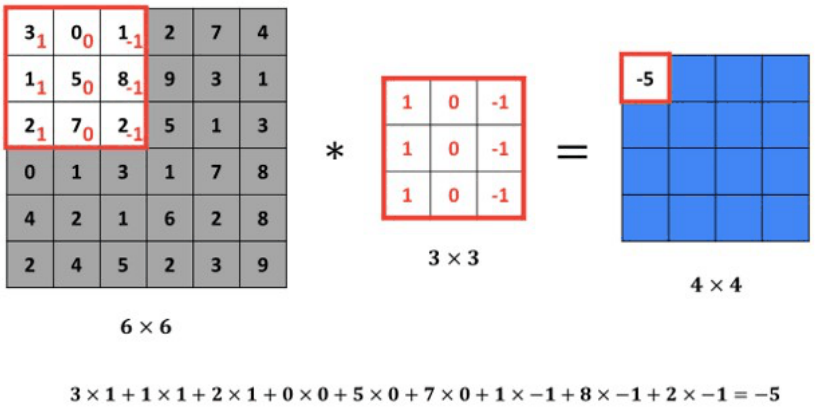


Figure 4.1: How a convolution operation works

Source: https://datahacker.rs/edge-detection/ [19]

The convolutional layer is the driver for feature mapping in a CNN. It applies a convolution operation shown in Figure 4.1 to the input data. The operation is done element by element while sliding a filter (also called kernel) over the input data. On each step, the Frobenius Product between the kernel and the submatrix given by the current position and the kernel dimension is calculated and noted in the output matrix. The parameters in the kernel matrix are chosen in such a way that the Frobenius Product is maximized when the kernel is over a feature that the kernel is supposed to detect. In Figure 4.1, the kernel for example is a vertical edge detector, meaning it will output a high amount (positive or negative) when the horizontal gradient in the input data submatrix has a high absolute value. This result is rather trivial, as a positive horizontal gradient as seen in the upper-left 3x3 submatrix of the example leads to a right column that when negatively weighted, overweights the positive-weighted left column and thus the output for the upper-left 3x3 submatrix is negative. Conversely, a Fresenius product of the upper-right 3x3 submatrix with the kernel returns a positive value, hinting at a negative horizontal gradient in the input data. The result of such a convolution can be observed in Figure 4.2.
In the context of weather data, the convolutional layer can be used to detect any weather patterns not just edges and the kernel itself can be learned by the network.

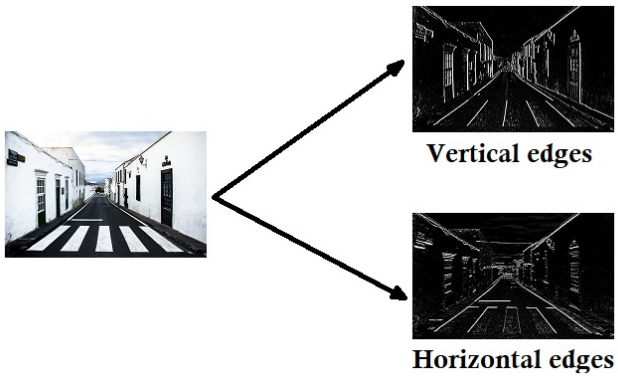
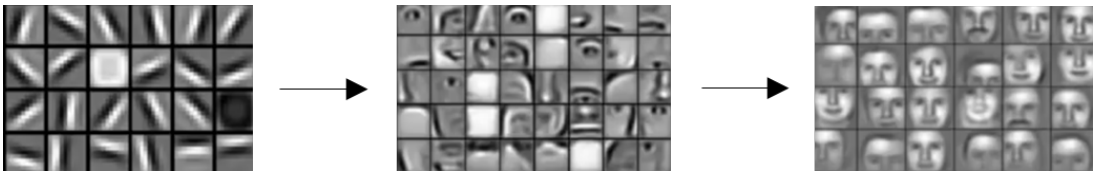


Figure 4.2: Example of edge detection with a convolutional kernel.

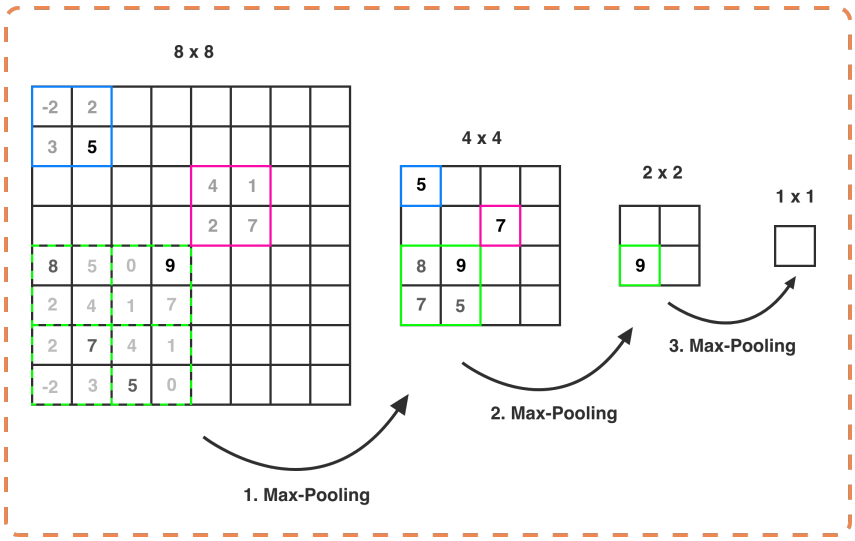
Source: https://datahacker.rs/edge-detection/ [19]

Activation Function

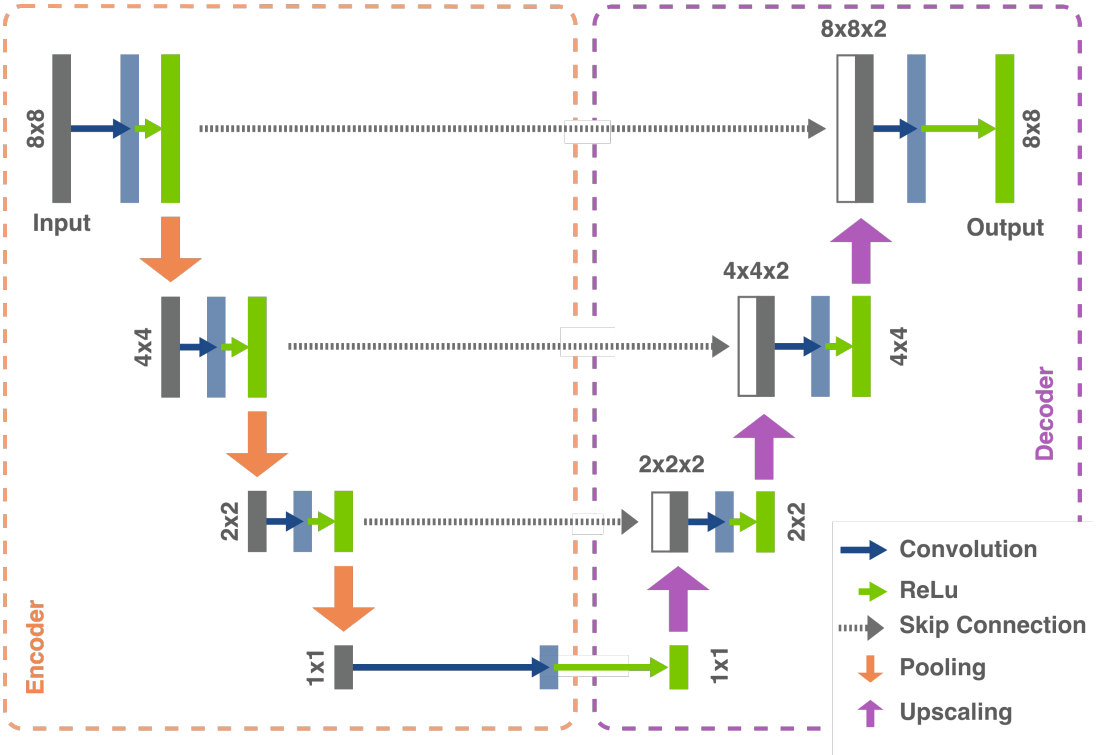
To map the output of the convolutional layer to a meaningful space, and to avoid negative values, an activation function is applied. This introduces non-linearity into the network. The simplest activation function is the ReLU function, which returns the input if it is positive and zero otherwise. It is defined as $f(x) = max(0, x)$. The ReLu function is the activation function used in this work after each convolutional operation.



(a) Low-level features are combined into high-level features.



(b) Illustration of 3 max pooling operations



(c) Encoder Decoder Architecture in the U-Net.

Pooling Layer

The exact position of a low-level feature in the input data is not so important when it comes to detecting high-level features, it is more important to recognize if a feature is present at all in area of the input data or not. Thus scaling down the resolution of the matrix by combining every 2x2 submatrix into one value can be beneficial. A possible way to aggregate them is by taking the maximum value of the submatrix, which will work for the mentioned purpose of detecting if a feature is present in the area of the submatrix or not. In Figure 4.1 the convolution operation itself also reduces the shape of the input data, which is because close to the borders of the input data the kernel can not be applied entirely, but in this work it is avoided by padding the input data. For the 8x8 grid cells of the ERA5 data, that is used as input in the laid out approach (see subsection 3.2), the architecture of the CNN could include a maximum of 3 pooling layers, reducing the input data to a 1x1 matrix as seen in Figure 4.3b. Figure 4.3b just illustrates the downsampling of the data, in the actual architecture, pooling layers always come together with convolutional layers and are never applied directly after each other as seen in the encoder part of Figure 4.3c. In this work, the pooling is integrated into the convolution operation by using a step size of 2 when sliding the kernel over the input data, reducing the shape of the data by half in each dimension.

Decoder

In the decoding phase of the U-Net architecture, the original data shape is reinstated by upscaling the encoded layers using detected features. Nearest-neighbor interpolation is employed for this purpose, wherein values in the matrix are replicated as required. As the decoder progressively restores the shape, it incorporates so-called skip connections with the encoder at each level. Skip connections entail copying the outputs of each encoding layer directly into the decoding process. This mechanism aids the network in retaining the contextual information of the input data, which proves particularly crucial in spatial infilling tasks, where the reconstruction must align with the available surrounding data [18]. To integrate the doubled data from skip connections with the upscaling layers, a convolution operation is applied once more (see Figure 4.3c), with the step size configured to halve the size. This combined architectural approach ensures that for any input data, the model generates a prediction of the same shape as the input, with the predicted values representing the missing data. The specific values are determined by the parameters employed in the convolution kernels.

Supervised Learning

With the correct convolution kernels found, the prediction should ideally be the same as the target data. The training process is devised to find the best kernel parameters, which are called weights in this context. To minimize any differences between ground through and prediction, the differences first need to be quantified in a loss function. As the model calculates the prediction hour by hour, we only need to compare the prediction to the measured temperature at the weather station at the same hour. However, as the prediction is a matrix and the target is a scalar, the absolute differences between the measured temperature and the prediction at each of the 64 data points in the output are added up.
Then the partial gradient of the loss function concerning each weight in the network is calculated. This involves recursively applying the chain rule of calculus, a process known as backpropagation. The weights are then adjusted in the opposite direction of the gradient to minimize the loss function. The magnitude of these adjustments is governed by the learning rate, a pivotal hyperparameter in the training process.

4.2 Atmospheric Reanalysis

Combining past weather observations with numerical model-based weather predictions (NWP) is a data assimilation technique called atmospheric reanalysis, particularly when applied over a long period with a consistent data assimilation system [20, 21]. The numerical models have degrees of freedom, allowing for adjustments of parameters, which are chosen in such a way that the model output matches the actual measurements wherever they are available in space and time, may it be from a weather station or a satellite.
An institution providing such reanalyses is the European Centre for Medium-Range Weather Forecasts (ECMWF), and over the years, they have produced several reanalyses, with the most recent one being ERA5 [22].
Several organizations besides ECMWF produce global atmospheric reanalyses. To name a few leading: MERRA-2 from NASA's Global Modeling and Assimilation Office (GMAO) [23], JRA-55 from the Japan Meteorological Agency (JMA) [24], and CFSR (version 2) from the National Centers for Environmental Prediction (NCEP) [25].
ECMWF began its reanalysis efforts with the FGGE project in 1979, followed by ERA-15 in the mid-1990s, ERA-40 in the early 2000s, ERA-Interim in 2008, and the most recent ERA5 reanalysis in 2017 [22]. ERA5, the fifth generation reanalysis from ECMWF, surpasses ERA-Interim in accuracy and completeness [22], which was previously considered the most accurate global reanalysis product [26]. This makes ERA5 the optimal choice

for input data in this work. ERA5's improved accuracy over ERA-Interim is particularly notable in East Africa, where weather stations are sparse and the 3D-printed weather stations aim to make a significant impact [27]. This justifies using ERA5 as a benchmark for comparing the accuracy of the reconstructions produced in this work.
The ERA5 data is available in grid cells of 0.25° x 0.25°, approximately 28 km x 28 km at the equator, and involves four-dimensional data assimilation, considering latitude, longitude, time, and multiple pressure levels in the atmosphere. The technique used for ERA5 is known as 4D-Var [9]. Although ERA5 offers numerous variables, this work utilizes only the 2-meter temperature as input for the neural network, corresponding to the measurements targeted by the 3D-printed weather stations. However, for similar or extended approaches, other variables such as wind components and solar radiation could also be considered.

5 Results

5.1 Stations included in this study

From the 3D-Paws project (see section 2), measurements from three different weather stations in recent years were obtained. The stations are located in Marshall, Colorado; Vienna, Austria; and Barbados, Caribbean.
The station in Marshall is located between Boulder and Denver, next to Marshall Lake, at an elevation of 1,743 meters. It is less than 10 km east of South Boulder Peak, which exceeds 2,600 meters in elevation [28]. Further, the station is situated only about 30 km east of the Colorado Front Range, where the Rocky Mountains range in altitude between 3,250 and 4,000 meters [29]. This topographical context is visible in the ERA5 data (Figure 5.2).
Vienna lies to the west within the Vienna Basin, at a relatively low altitude just beyond the Alpine region. The Alps, which rise moderately to the west of Vienna, reach heights of up to 2,000 meters above sea level within approximately 100 km. The ERA5 resolution is sufficient to differentiate atmospheric conditions and topographical features in this region. The station itself resides in the north

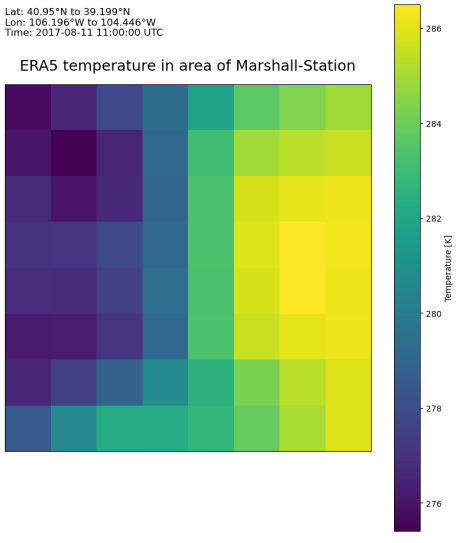
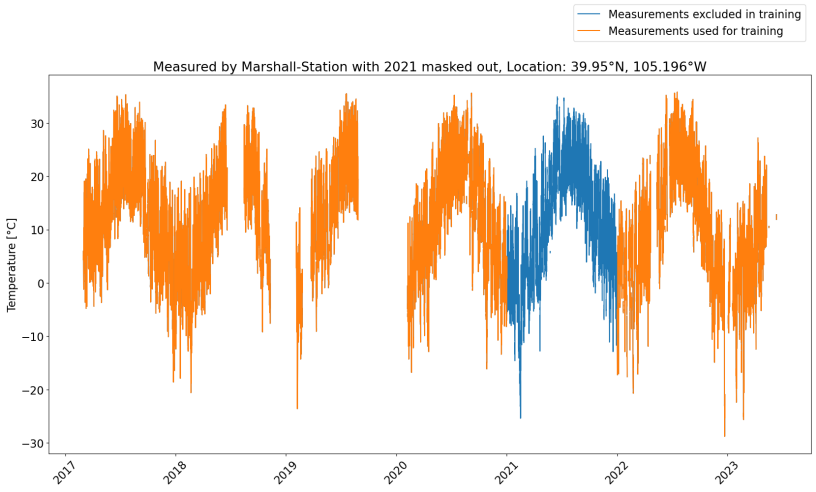
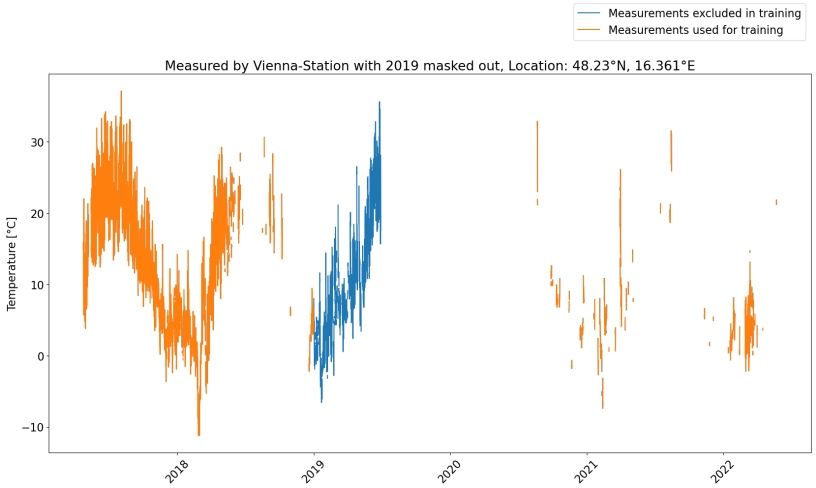


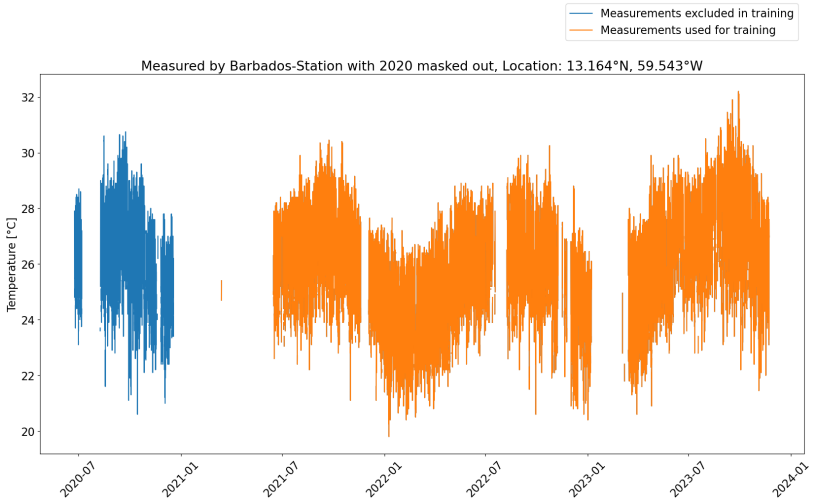
Figure 5.2: ERA5 Area of the station in Marshall, Colorado



(a) Station in Marshall, Colorado, USA



(b) Station in Vienna, Austria



(c) Station in Barbados, Caribbean

Figure 5.1: Available data from 3D-printed Weather stations

of the city on university grounds at an altitude of 159 meters. East of it is the urban Danube Canal, separated from the station by only a main road and a railway line.
The station in Barbados stands in the parish of St. George at an altitude of 274 meters. The island is relatively flat, with its highest point being Mount Hillaby at 340 meters. The island's size, approximately 34 km long and 23 km wide, is similar to the size of an ERA5 grid cell. As observed in Figure 3.1, surrounding grid cells are predominantly oceanic. This means the ERA5 data cannot accurately capture the diurnal cycle of the station, which is influenced by its land location while surrounding cells are primarily oceanic.

5.2 Data Availability

The measurements span up to late 2023, with the quantity of available temperature data varying significantly between stations. The station in Marshall, one of the oldest in the 3D-Paws project, has the most available measurements among the three stations. The weather station in Vienna has data dating back to 2017, but it has been almost entirely offline since mid-2019. The weather station in Barbados began recording in mid-2020.
Figure 5.1 shows the measurements from the three stations after data cleansing and conversion to an hourly basis. The measurements were provided by the US National Center for Atmospheric Research (NCAR), with most invalid values already marked. However, especially in Barbados, the sensors had more noise and occasional invalid measurements, requiring extra cleansing. Temperatures near zero and below were excluded. The conversion from minute to hourly data was done by averaging all minute values in each hour, provided there were more than 20 values available and they were not all the same. It was observed that the sensors sometimes got stuck, delivering the same value for extended periods.
The dataset for the Marshall station spans from 2017 to 2023, comprising 41,883 data points. These measurements reveal three significant data gaps, with larger intervals without data occurring in the midsummer period of 2018, late 2019 to early 2020, and the most extensive gap from September 2019 to January 2020.
For the Vienna station, measurements are available for only 12,477 hours, less than a third compared to Marshall. This is primarily due to extended downtimes starting from mid-2018, with few to no measurements except for a brief period from late 2018 to July 2019. After that, there were only sporadic data collection instances before the station went offline in April 2022.
The Barbados station had two significant downtimes longer than a month, the largest being the first half of 2021 and the second being the first quarter of 2023. In total, the

station has 17,315 hours of measurements, still less than half of the Marshall station.

Splitting into Training and Validation Data

As explained in section 3, the measurements are split into a training and validation dataset. The training dataset is used to train the model, while the validation dataset will be kept aside to compare it to what the model predicts after the training to validate if and to which extent the model learned to reconstruct the missing data. If the validation data would be included in the training data, the model would be able to predict the values it has already seen, but it would be impossible to tell if the model learned to generalize the local weather patterns. Starting with the Marshall station, the data was sufficient to extract an entire year as validation data. Excluding a consecutive year from the training data not only allows for a comprehensive analysis of a full yearly cycle but also ensures that the model relies solely on the general weather patterns it learned from the training data to predict the values of the validation data. This approach prevents the model from potentially "cheating" by accessing information from nearby training data points, which could compromise the integrity of the validation process. It is a common practice in the machine learning field to validate data as a contiguous set, as it helps to maintain the independence and integrity of the validation process. 2021 highlighted in blue in 5.1a was the most complete year in the Marshall dataset thus it was chosen as the validation data. With the mask of 2021 applied the training data for Marshall consists of 34,188 hours of measurements which is about 82% of the total data.
For the Vienna station, the data was split into training and validation data based on the availability of the data. The training data consists of all available data up to 2019, while the validation data is the year 2019. Resulting in 9,593 hours of measurements used as training data, which is about 77% of the total data. The limited availability of measurements didn't allow for a full year of validation data.
The Barbados station also only had 2022 and 2023 as a complete available years, so that the decision to use the available data of 2020 as a training was a result of the lack of data as well. With the mask applied, the training data consists of 14,576 hours of measurements, which is about 84% of the total data.

Execution of Training

For the training and validation data, the corresponding temperature data of the 64 ERA5 grid cells in the regions of each station was obtained. The 8x8 grid cells were chosen to be centered as well as possible around each weather station, given the specific ERA5

Station	Total Data	Training Data	Validation Data	Validation Ratio
Marshall	41,883	34,188	7,695	0.18
Barbados	17,315	14,576	2,739	0.16
Vienna	12,477	9,593	2,884	0.23

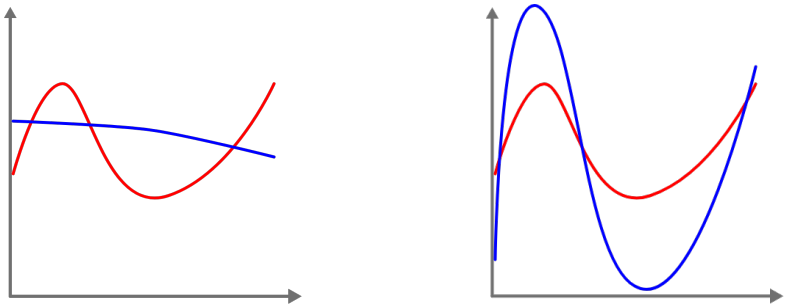
Table 1: Data availability and split into training and validation data

coordinates. The ERA5 data was also preprocessed using the code pipeline described in section 6. Then, the CNN with the architecture described in subsection 4.1 was trained on the training data. The training was done in batches of 4 data points, which are, in this case, four hourly steps at a time. The model was trained for up to 300,000 iterations, which, therefore, depend on the dataset size between 8 and 31 epochs. The training was done on the Supercomputer Levante of the German Climate Computing Center, which made it possible to complete training runs in just a few hours. However, the training can be done in the same way on almost any machine. During the training, in the scientific process, where the approach is un-proven to work, it is crucial to validate multiple times during the training to calculate not only the loss function over the model for the training data, but over the validation data. This is done to prevent overfitting, a so-called phenomenon where the model in the aim to minimize the loss for the training data, which is what training using backpropagation is exactly at it is core, starts to learn the noise and any potential specifics of the training data so well, that it loses it is ability to generalize and the loss for the validation data starts to increase. This is a sign that the model is overfitting and the training should be stopped. The model that is then chosen is the one that performed best on the validation data. To allow for that, not only was the ERA5 data for the training data provided to the machine learning architecture but the validation ERA5 data was also paired with the expected output for these, the masked measurements of the weather station.

Training runs have been completed for all stations with up to 1 million iterations, but the best models were in most cases, already found between 300 and 500 thousand iterations. The models were then applied to the ERA5 data at all the hours in the validation data set of each station, to analyze the performance of the model for each station.

5.3 Validation Methods

As numerical measures of quality, the Root Mean Squared Error (RMSE) and the correlation coefficient were calculated. RMSE quantifies proximity between two series in terms of distance (refer to Figure 5.3a), yet it lacks insight into their trend relationship. Conversely,



(a) Low RMSE, low Correlation Coefficient

(b) High Correlation Coefficient, high RMSE

Figure 5.3: Comparison of the RMSE and Correlation Coefficient

the correlation coefficient gauges the model's fidelity to the overall trend of the observed data (as depicted in Figure 5.3b). Notably, high correlation can coexist with significant distance between series.

Root Mean Squared Error (RMSE)

To quantify how far apart the reconstructed temperature series $\hat{y}$ is from the measured temperature series y, where both series have n different timesteps and thus can be seen as vectors, most commonly the root mean squared error (RMSE) comes to help. The RMSE is strongly connected to the Euclidean norm of the difference between both vectors $\hat{y} - y$. However, a simple mean absolute error (MAE), also called the Manhattan norm, could seem most intuitive and easiest to understand. It would be calculated just by adding all absolute differences up before dividing it by the length of the series. The RMSE is favored in machine learning and statistics because it penalizes larger differences more than smaller ones, which is more in line with the human perception of errors.

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

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2}$$

When converting Equation 1, it can be seen how the RMSE is connected to the Euclidean norm:

$$\text{RMSE} \cdot \sqrt{n} = \sqrt{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} = \|y - \hat{y}\|_2 \tag{3}$$

The Euclidean norm $\|y - \hat{y}\|_2$, of a coordinate, in our case, the error $\hat{y} - y$ can be imagined as the diagonal distance between the origin and the point with these coordinates. For demonstration purposes, a right triangle in 2-dimensional space can be imagined with the errors at two validation timesteps as the legs. The RMSE over both errors would be proportional to the hypothenuse of that triangle which is more dependent on the length of the longer leg than the shorter one. The Manhattan norm (Equation 2), on the other hand, which received its name from the gridded street system of Manhattan, is not based on the diagonal distance but the distance a taxicab in Manhattan would have to drive to travel along the hypothenuse, the equally weighted sum of the legs.

Pearson Correlation Coefficient

The Pearson correlation coefficient (which in the following is just called correlation coefficient), is a measure of the strength and direction of a linear relationship between two variables. It ranges from -1 to 1, where 1 indicates a perfect positive linear relationship, -1 is a perfect negative linear relationship, and 0 is no linear relationship. The correlation coefficient is calculated as follows:

$$\text{Correlation Coefficient} = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2 \sum_{i=1}^{n}(\hat{y}_i - \bar{\hat{y}})^2}} \tag{4}$$

Where $\bar{y}$ and $\bar{\hat{y}}$ are the mean values of the measured and the model output temperature series, respectively. The correlation coefficient is a measure of how well the model output follows the general trend of the measured data. [30]

5.4 Validation of the Models

Marshall station

Beginning with Marshall, where the most abundant training data is available, the model's predicted temperatures for the hourly measurements of the validation dataset are depicted in Figure 5.4 by a blue line against the ground truth in black. Given the high number of hourly timesteps along the x-axis spanning the year 2021, for the sake of readability, the temperature differentials are plotted on the y-axis instead of the absolute temperatures. Above the chart, the RMSE and correlation coefficient of the reconstructed series are provided for the corresponding data. The RMSE for the Marshall station is 2.02°C, which is the highest value among the three weather stations. The plot depicts in red the temperature at the nearest ERA5 grid point (located at coordinates 39.950° N, 106.196°

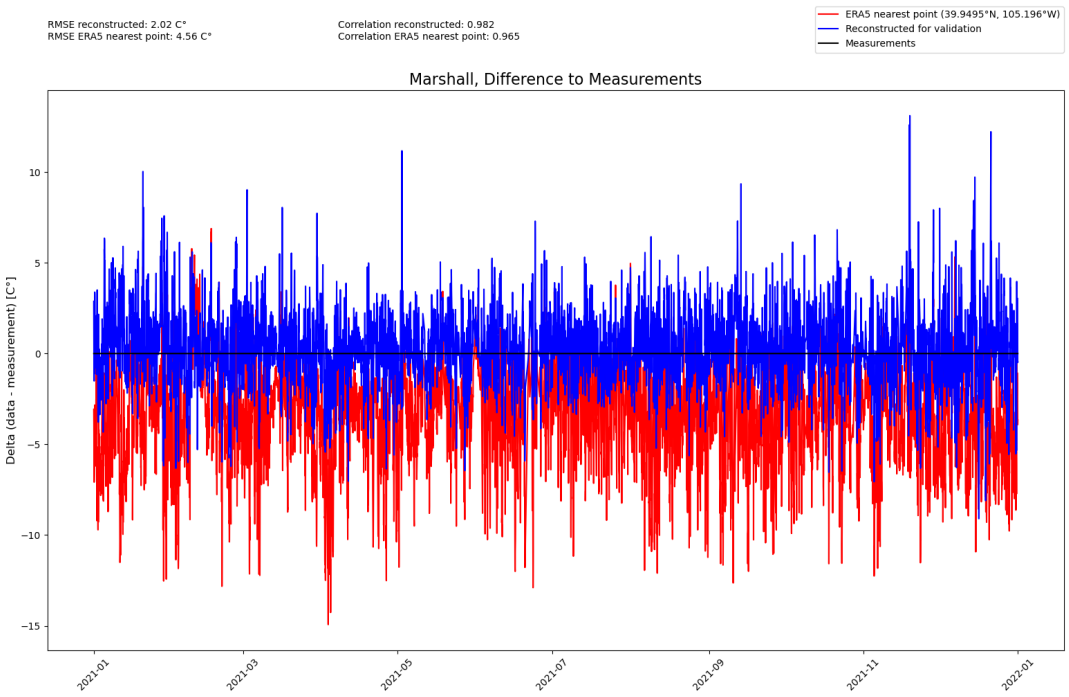


Figure 5.4: Difference between reconstructed and measured hourly temperature for Marshall station

W) for each timestep to assess the discrepancy relative to a direct readout from ERA5. The grid point's location is precisely the location of the weather station in Marshall (see Figure 5.1a). However, the RMSE between the measurements and the temperature time series of that ERA5 grid point is 4.56°C, which is more than twice as high. Considering that the model was trained on the ERA5 data of the 8x8 grid cells around the station, the model was able to reduce the error significantly. The ERA5 data, as seen in the chart, has a substantially low bias compared to the station, which directly relates to the fact that the grid cell of the station already reaches into the Rocky Mountains, as described above. Thus, it is especially important that the reconstruction not only has a lower RMSE, which could be achieved in this case just by correcting the low bias, but also beats it is training data in terms of correlation. The calculated correlation coefficient over the validation year 2021 on an hourly basis for the reconstructed temperature and the measurements is 0.982 compared to 0.965 for the ERA5 nearest grid point, a significant improvement.
Even better is the performance of the model when the hourly values are combined again with daily means, as in Figure 5.5. The chart shows the daily mean temperature of the three different series, known from the previous chart over 2021. In this case, the reduced number of time steps allows for an absolute temperature scale on the y-axis, which makes

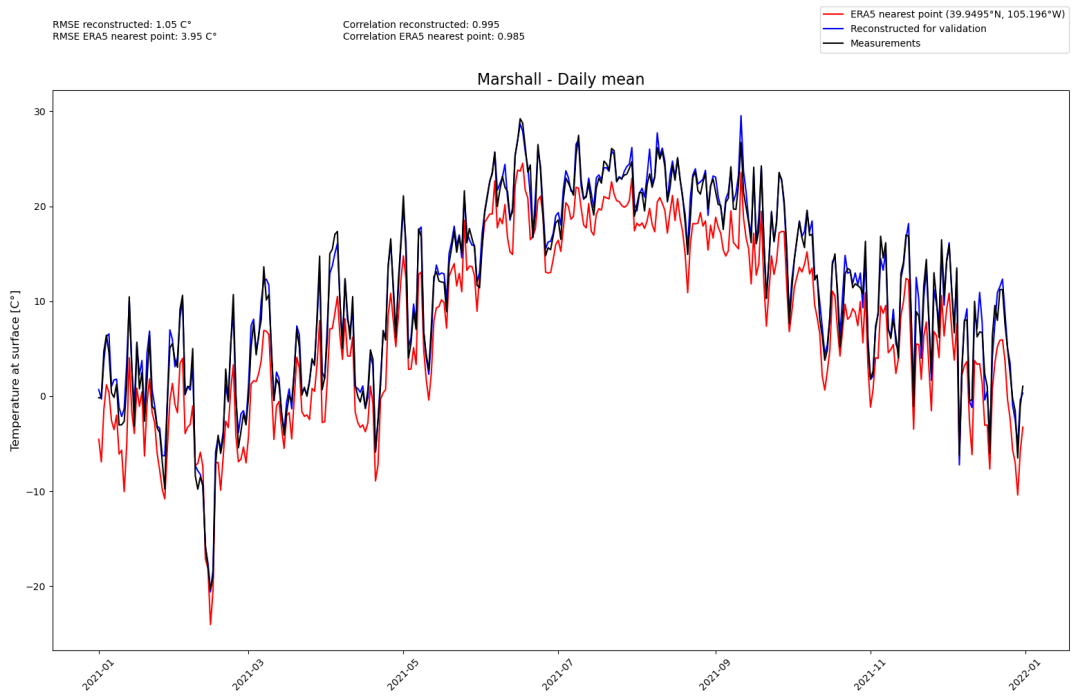


Figure 5.5: Reconstructed temperature vs measured temperature for Marshall station (Daily mean)

the performance of the model visually appealing. The RMSE on a daily basis for the prediction is almost halved the original one at 1.05°C, while the correlation coefficient is even higher as well at 0.995, the ERA5 correlation coefficient improved significantly as well; however, the RMSE did not as the ERA5 data was impacted drastically by the low-bias.

Figure 5.6 makes clear that the improvement in RMSE and correlation coefficient of the daily analysis was not caused by potential issues in the model to predict the diurnal cycle correctly but by noise reduction in general. In Figure 5.6 the average diurnal cycle of the reconstructed temperature is plotted against the measured temperature and the temperature series of ERA5. This means that over all the days in 2021, for each of the 24 hours, an average temperature was calculated and plotted for the three series. As all measurements have been recorded in universal time code (UTC), which is also the time in ERA5, the x-axis also uses UTC, which is apart from the daylight savings time 6 hours ahead of Colorado. It can be observed that ERA5 has a higher amplitude in the diurnal cycle than the weather station and that the model is capable of correcting that.

It is not surprising, that the RMSE and correlation coefficient improve the more the data is aggregated, so it is also essential to look at the hourly values again and go into detail.

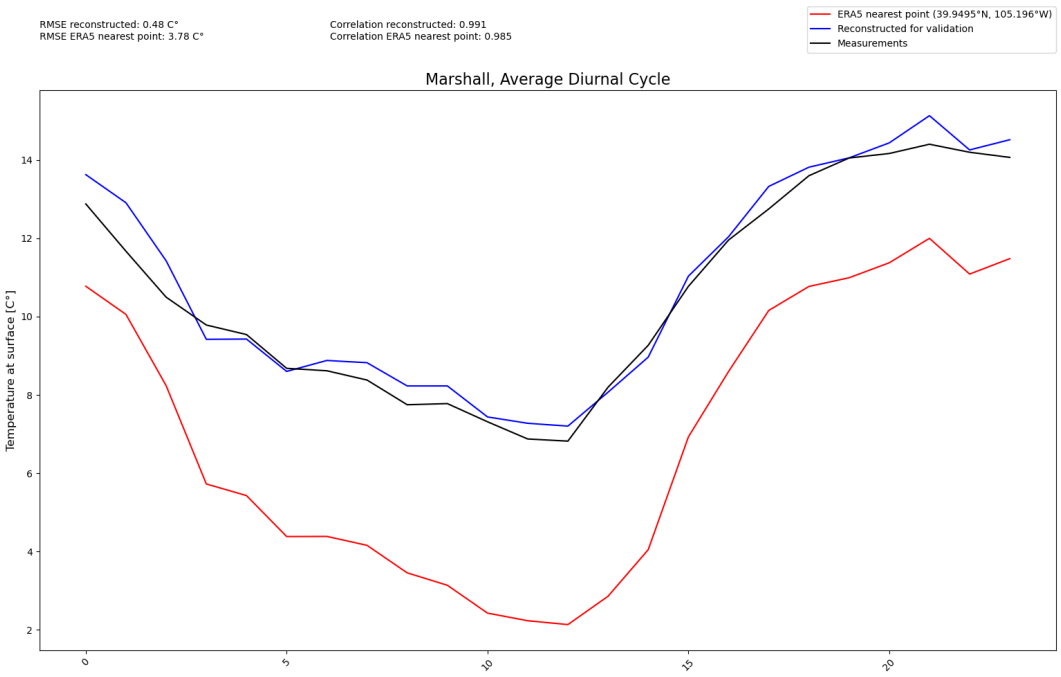


Figure 5.6: Reconstructed temperature vs. measured temperature for Marshall station (Average Diurnal Cycle)

To allow the details to appear on a plot, the x-axis can be cropped to a shorter period, as in Figure 5.7, where the x-axis includes 168 hours, equivalent to 7 days. The chart is chosen to be representative, while some weeks have a better fitting than others. In this case, it can be seen that even in the measurements of 2021, minor gaps happened. For example

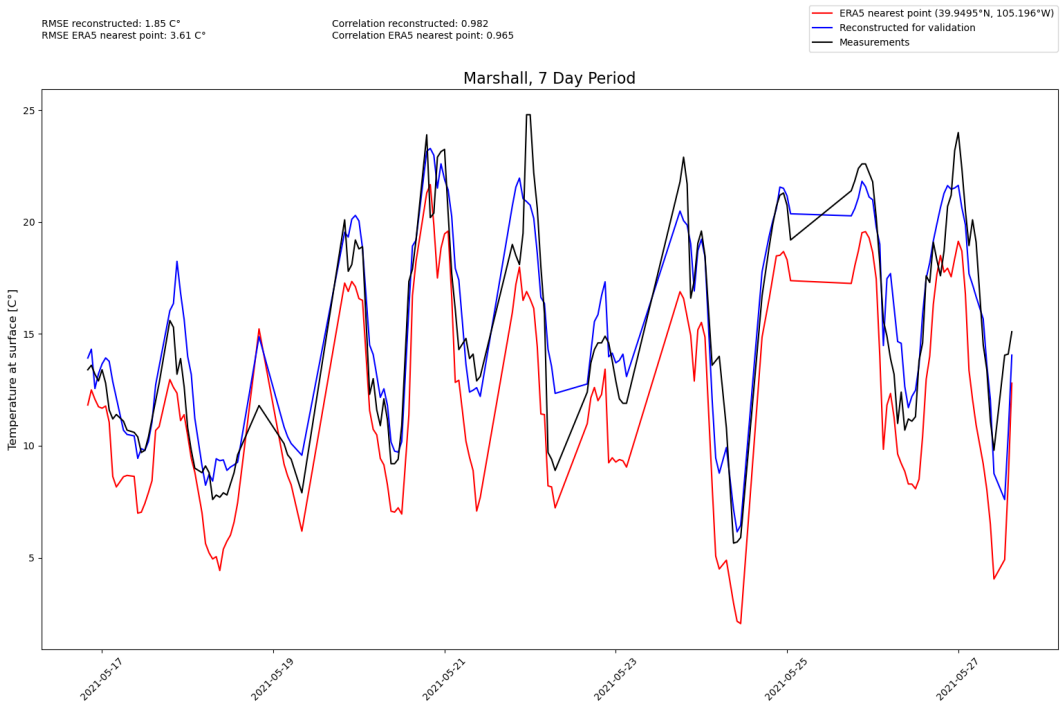


Figure 5.7: Reconstructed temperature vs. measured hourly temperature for Marshall station (7 Day Period)

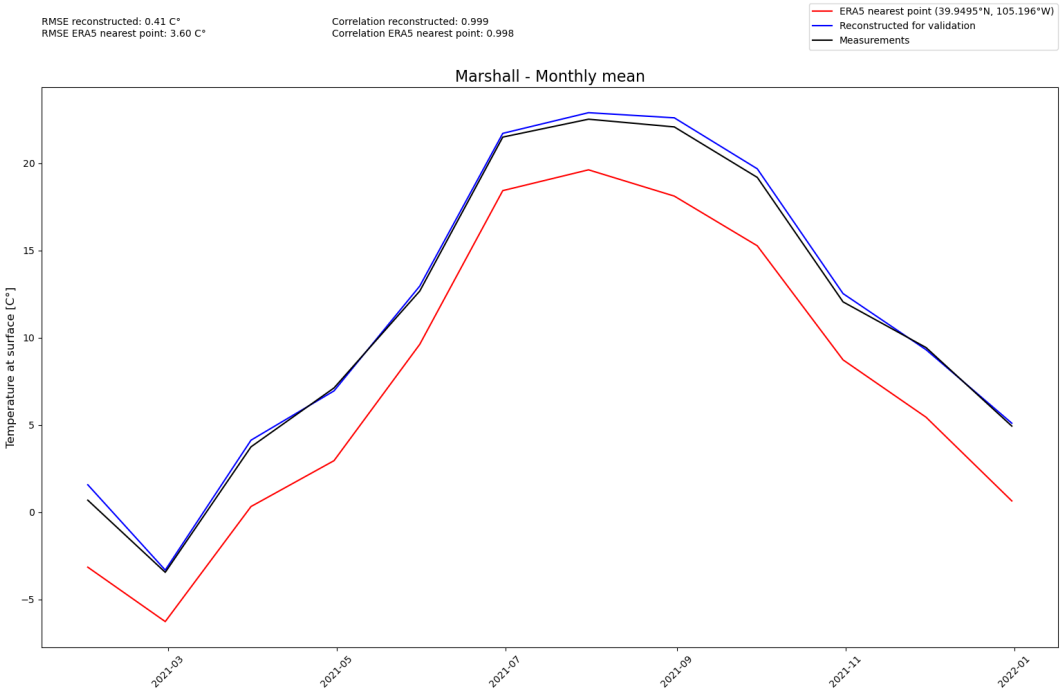


Figure 5.8: Reconstructed temperature vs. measured temperature for Marshall station (Monthly mean)

Vienna station

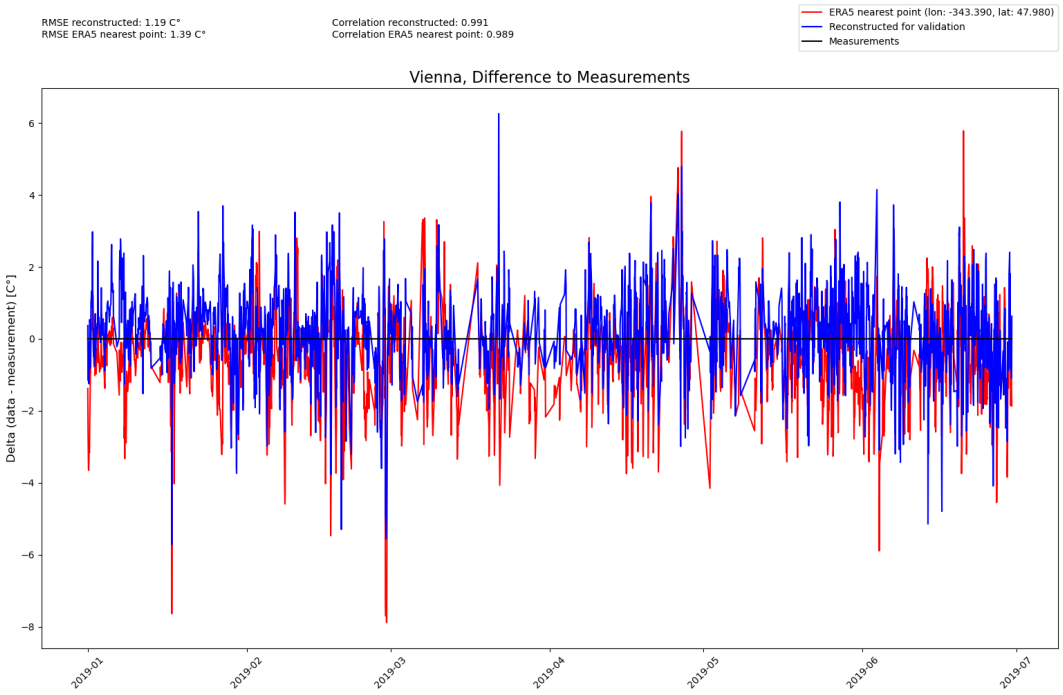


Figure 5.9: Difference between reconstructed and measured hourly temperature for Vienna station

Similar to the Marshall station, the plots for the Vienna station follow the same structure. Again, a plot over the full validation period on an hourly basis (Figure 5.9), daily (Figure 5.10) and monthly (Figure 5.13) aggregated ones and the diurnal cycle (Figure 5.11) as well as an extract of a week (Figure 5.12), are shown. However, a significant difference is that the station's available data in Vienna was so limited that only half a year was included in the validation data. Also, to put the validation of the reconstruction in perspective, it needs to be considered that the RMSE and correlation coefficient, in relation to the ERA5 data in Vienna, are by far the best among the three stations. 1.19°C RMSE and 0.991 correlation coefficient for the reconstructed temperature compared to 1.39°C RMSE and 0.989 correlation coefficient for the ERA5 data at the nearest grid point. The model was able to reduce the error slightly but not significantly. This minor improvement can be attributed to the limited training data but also to the fact that the ERA5 data is already very similar to the station data.

Aggregating the hourly values to daily means, as shown in Figure 5.10, unsurprisingly improves the RMSE and correlation coefficient again and allows for an absolute temperature scale on the y-axis. It can be seen that the temperature rose in Vienna from January to

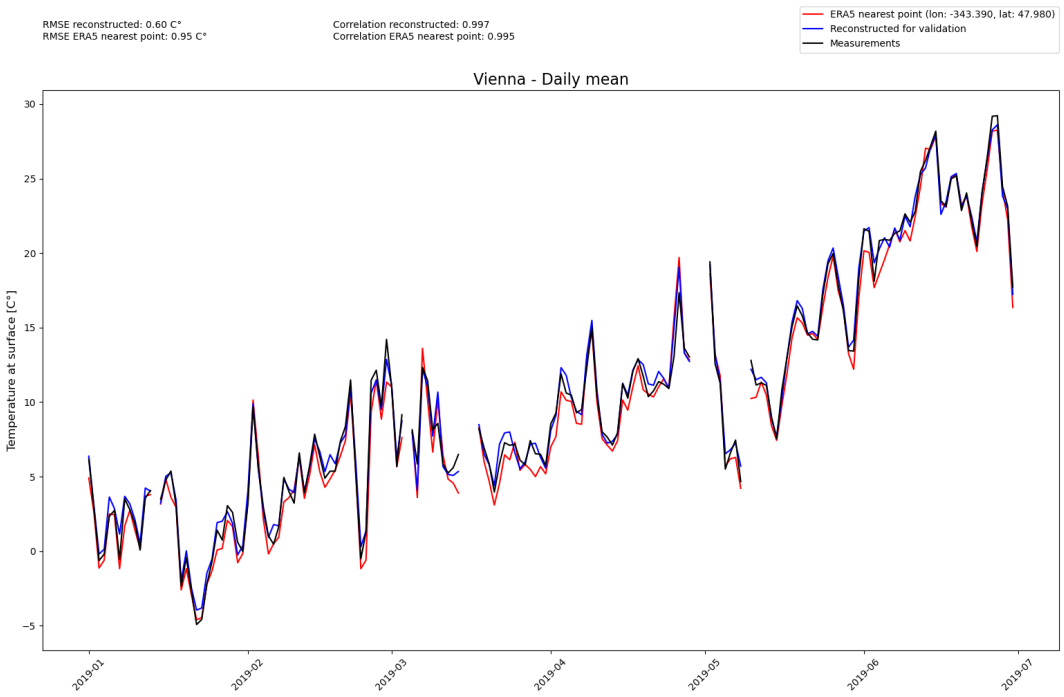


Figure 5.10: Reconstructed temperature for Vienna station (Daily mean)

July until the station went offline. By looking at the correlation coefficient, it seems that the reconstruction is just marginally improving over its input data. However, the RMSE at 0.60°C is, by a relative measure, significantly lower than the 0.95°C of the ERA5 data at the nearest grid point.

Figure 5.11 of the diurnal cycle shows the best performance of the model in Vienna, with an RMSE of 0.33°C compared to 0.73°C for the ERA5 data. The correlation coefficient of 0.968°C for the ERA5 data is the lowest among the different analyses for Vienna, which is due to the fact that there seems to be a time shift. As the temperature rises at the beginning of the day, the station temperature seems to be approximately one hour ahead of the ERA5 temperature. The model was able to correct this time shift, leading to this excellent RMSE and correlation coefficient of 0.985.

When aggregating to a monthly mean, it becomes further visible how close the ERA5 and reconstructed temperature are to the measurements. The correlation coefficients are both 1.000, indicating a perfect correlation. The only issue visible is that ERA5 has a slightly low bias, which the model minimally overcorrects.

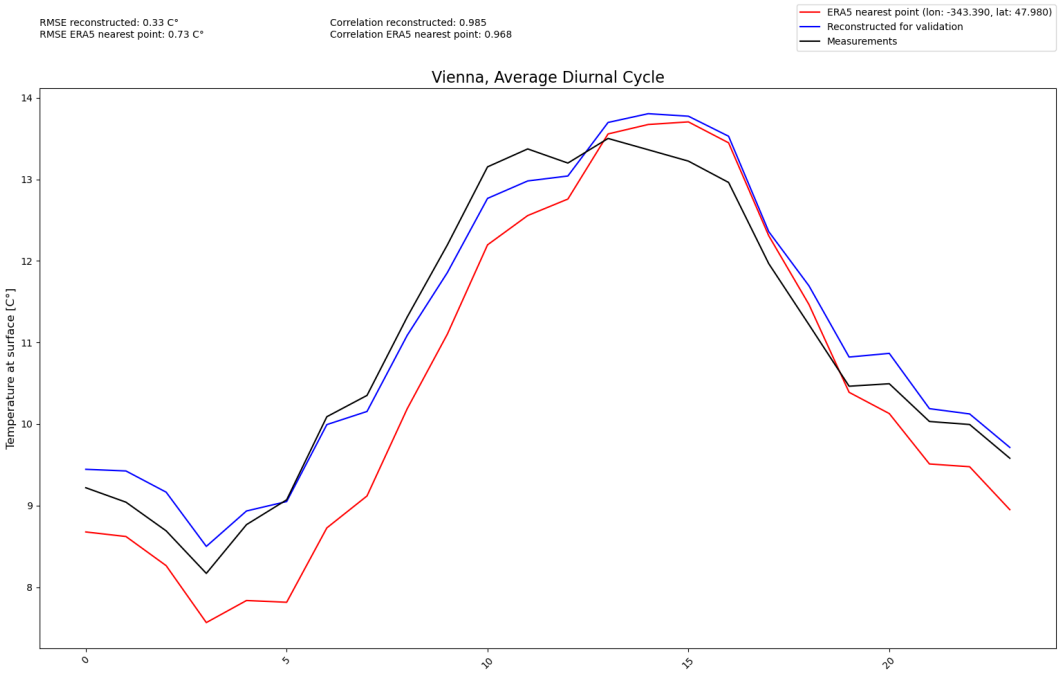


Figure 5.11: Measured temperature for Vienna station (Average Diurnal Cycle)

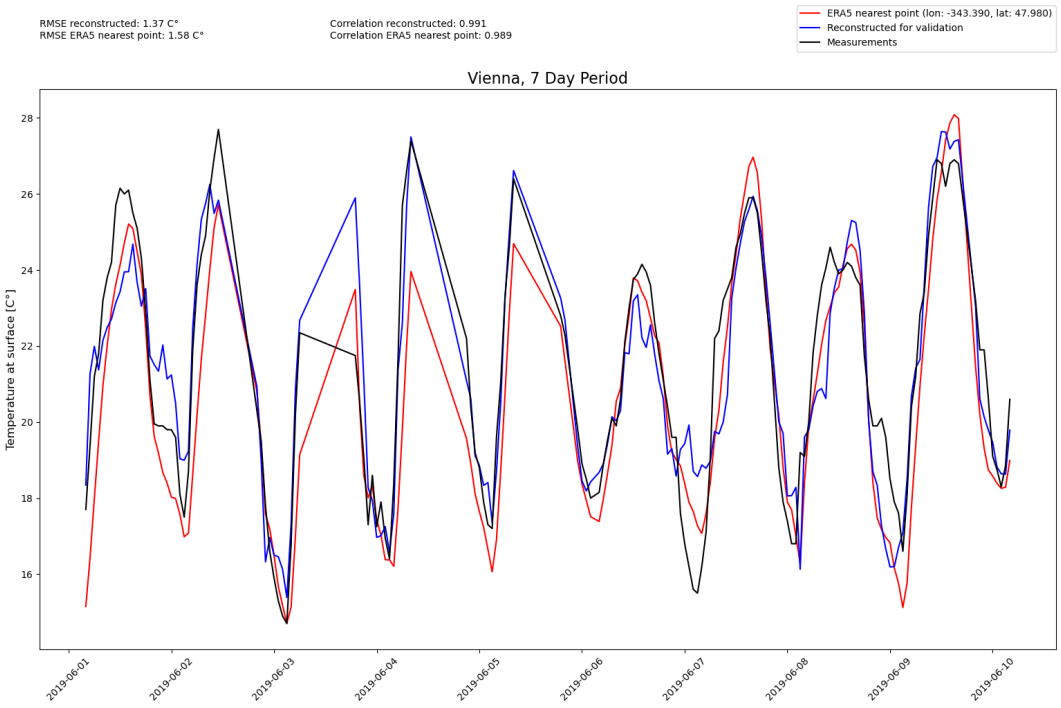


Figure 5.12: Reconstructed temperature vs. measured hourly temperature for Vienna station (7 Day Period)

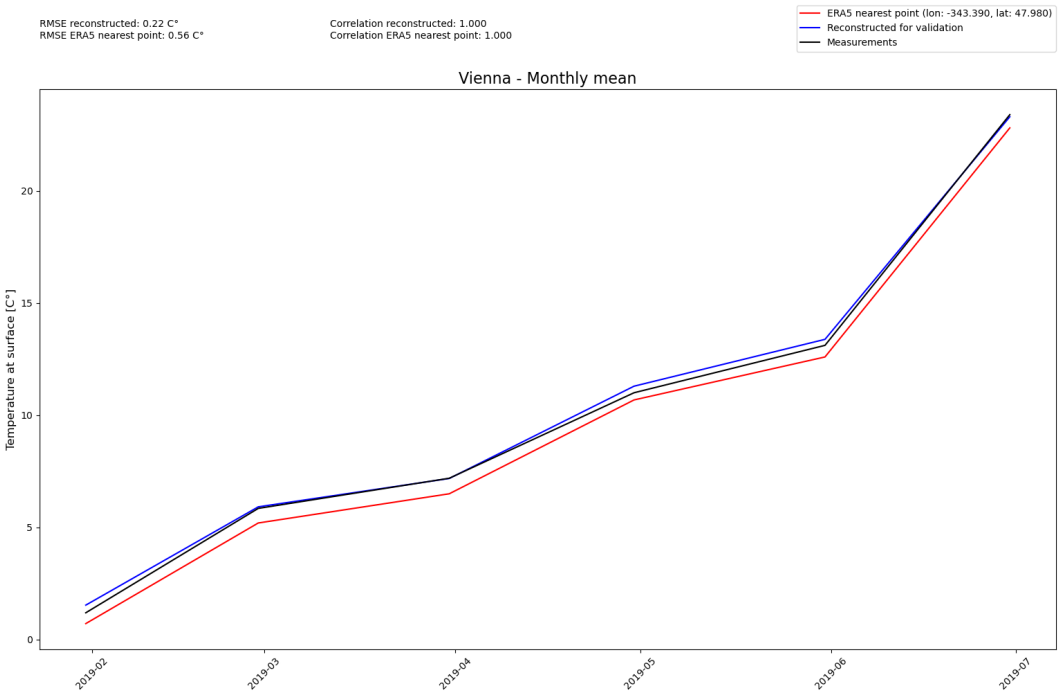


Figure 5.13: Reconstructed temperature vs. measured temperature for Vienna station (Monthly mean)

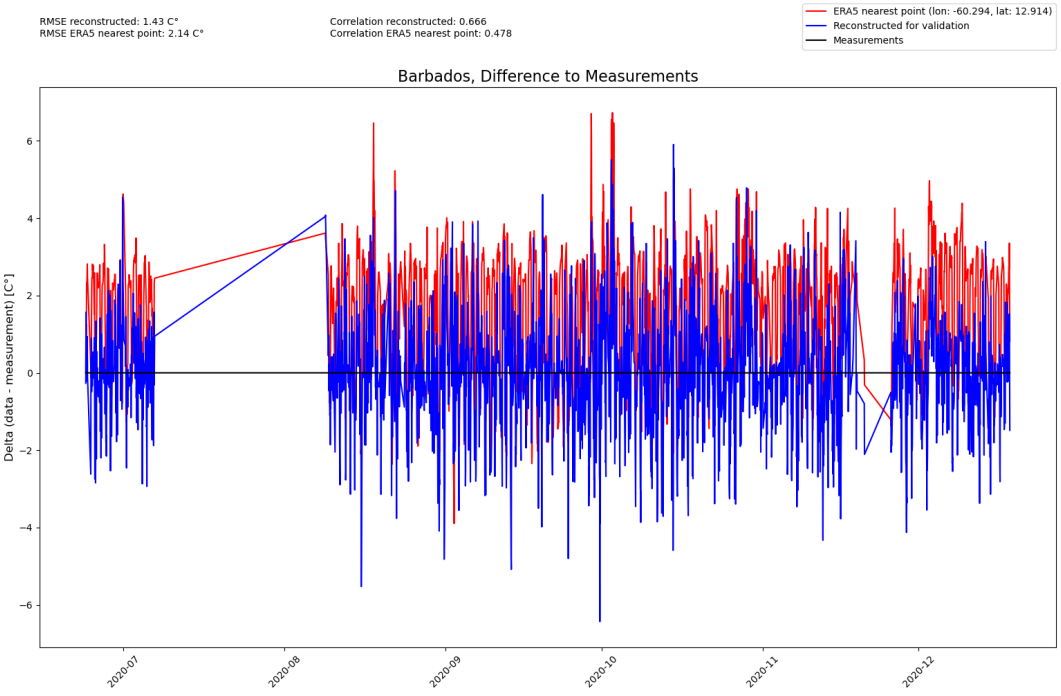


Figure 5.14: Difference between reconstructed and measured hourly temperature for Barbados station

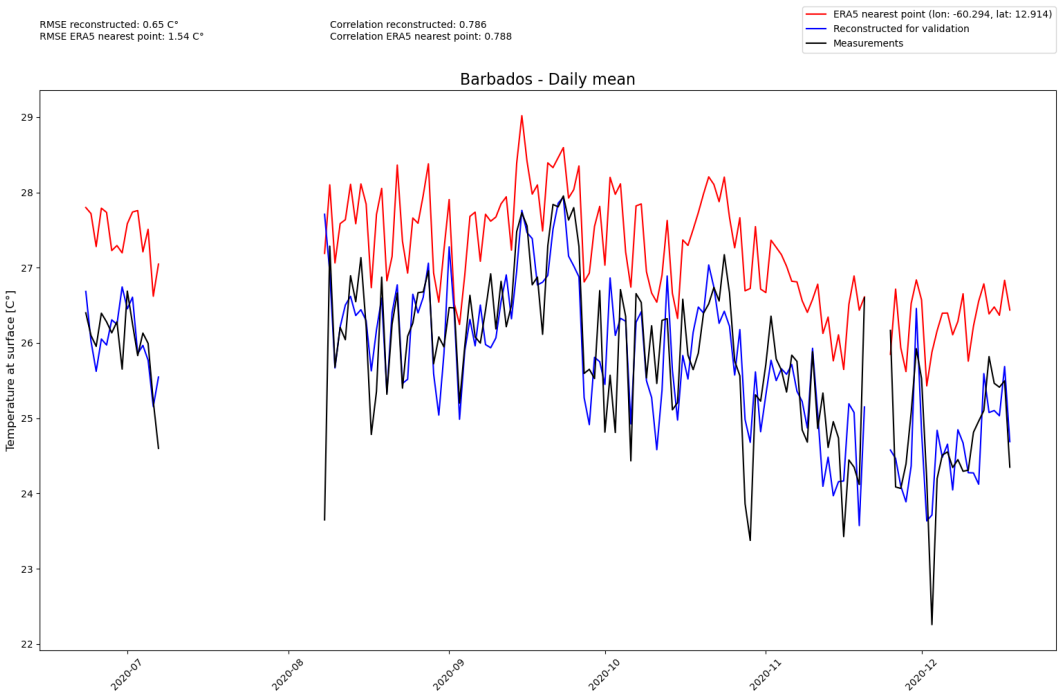


Figure 5.15: Reconstructed temperature for Barbados station (Daily mean)

Barbados station

Lastly, for the Barbados station, the model was able to improve the RMSE from 2.14°C for the ERA5 data to 1.43°C for the reconstructed hourly temperature over the full validation data (Figure 5.14). The ERA5 data had a very low correlation coefficient of 0.478, which is by far the lowest among the three stations.

In Figure 5.15, the daily mean temperature is plotted for the Barbados station. For the ERA5 data, the RMSE did not improve as much as the correlation coefficient, which is now at 0.788, while the RMSE is at 1.54°C. This indicates that the low correlation coefficient of the ERA5 data is mainly due to the diurnal cycle, which is not captured by the ERA5 data.

The diurnal cycle in Barbados is the most significant difference between the ERA5 data and the measurements. As mentioned previously, the ERA5 grid cells near the station are predominantly oceanic. This means that the diurnal cycle more represents the 2-meter temperature over the ocean than the land, which does not pick up a significant temperature amplitude over the day. However, the model was able to reconstruct the diurnal cycle of the station to a point where the correlation coefficient is really high again, at 0.964. In contrast, the ERA5 data has a correlation coefficient of 0.482. The RMSE of the reconstructed diurnal cycle temperature is 0.55°C, while the RMSE of the ERA5

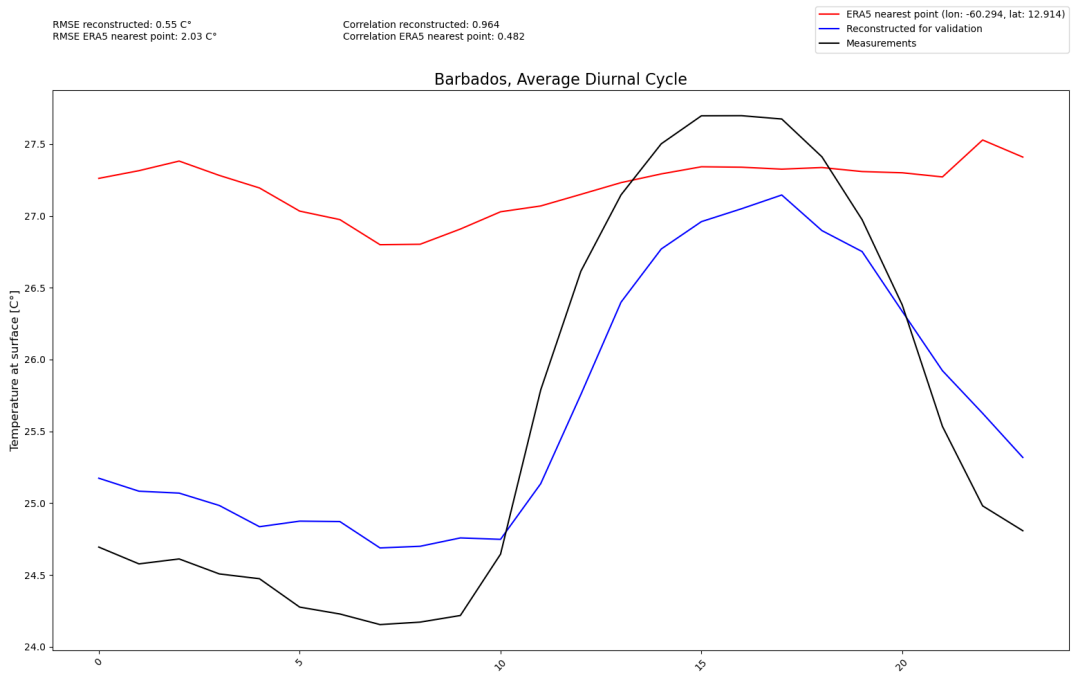


Figure 5.16: Measured temperature for Barbados station (Average Diurnal Cycle)

data is 2.03°C. This is a significant improvement, but the amplitude of the diurnal cycle is not as high as in the measurements. The correlation coefficient is still high, as it is a measure of the linear relationship between the two series, which is still given.

Summary

The results indicate that the CNN-based model significantly improves the accuracy of temperature data reconstruction compared to the ERA5 reanalysis dataset. This improvement is particularly notable for stations located in complex topographical areas. The model demonstrates its ability to generalize local weather patterns effectively, as evidenced by the substantial reduction in RMSE and high correlation coefficients across all stations. The Marshall station, situated in the Rocky Mountains region, showcases the model's capability to capture intricate topographical influences. While the Vienna station had limited training data, the model still managed to slightly outperform the ERA5 data, highlighting its ability to learn from smaller datasets, albeit with less pronounced improvements. For the Barbados station, the model excelled in reconstructing the diurnal cycle, which, due to the oceanic grid cells, was not clearly represented in the ERA5 data that it took as input.

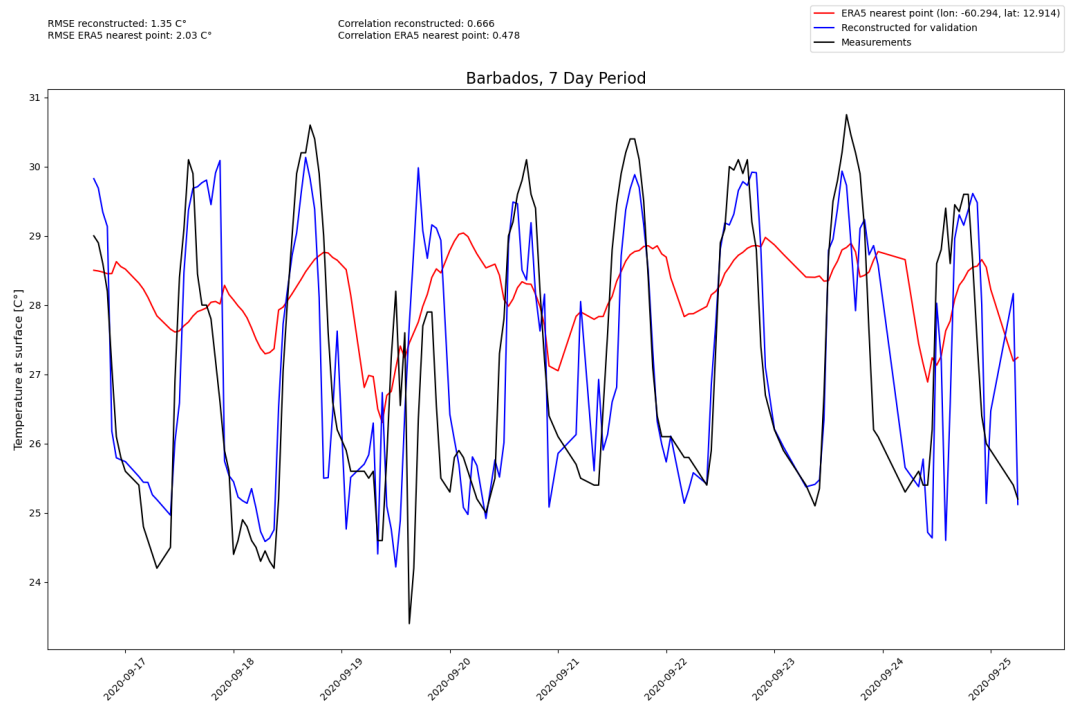


Figure 5.17: Reconstructed temperature vs. measured hourly temperature for Barbados station (7 Day Period)

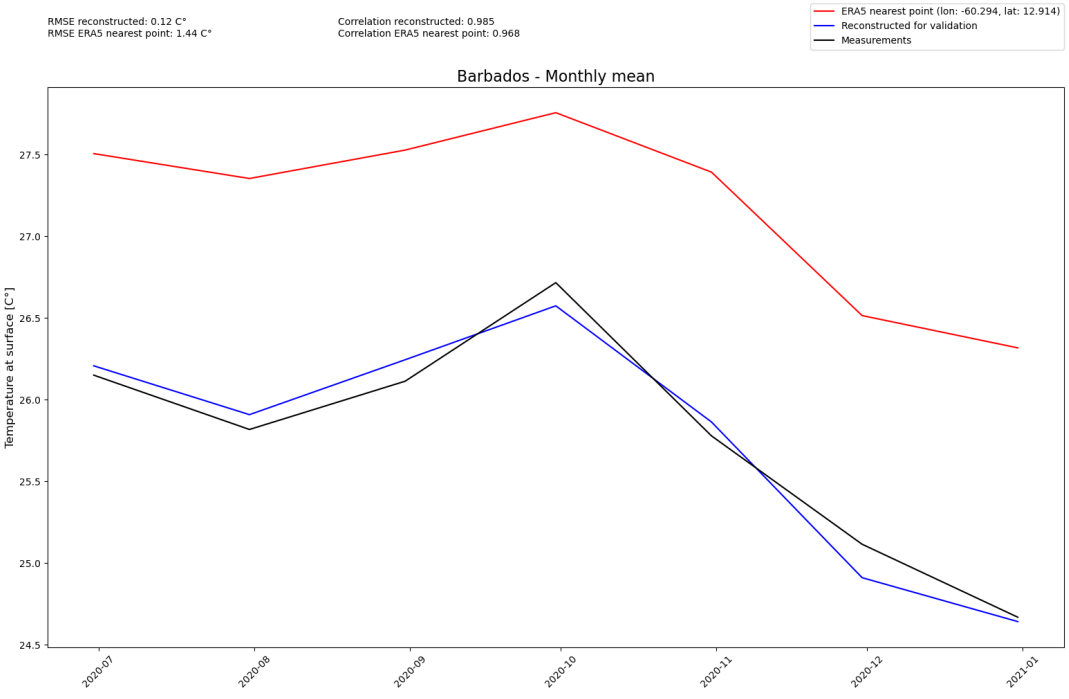


Figure 5.18: Reconstructed temperature vs. measured temperature for Barbados station (Monthly mean)

Station	Data	RMSE	RMSE ERA5	Correlation	Correlation ERA5
Marshall	Hourly	2.02	4.56	0.982	0.965
	Daily	1.05	2.02	0.995	0.982
	Diurnal	0.48	3.78	0.991	0.985
	Monthly	0.41	3.60	0.999	0.998
Vienna	Hourly	1.19	1.39	0.991	0.989
	Daily	0.60	0.95	0.997	0.995
	Diurnal	0.33	0.73	0.985	0.968
	Monthly	0.22	0.56	1.000	1.000
Barbados	Hourly	1.43	2.14	0.666	0.478
	Daily	0.65	1.54	0.786	0.788
	Diurnal	0.55	2.03	0.964	0.482
	Monthly	0.12	1.44	0.985	0.968

Table 2: Summary of RMSE and Correlation Coefficient for ERA5 and reconstructed data

6 Software Implementation

6.1 Introduction

Given that the machine learning part of the software is modulized as "Climate Reconstruction AI" (CRAI), the neural net's setup, training and evaluation are outsourced. Thus, the process for a single specific dataset of one station could be written pretty straightforwardly with a `NetCDF` file of the station data at hand if there is sufficient access to ERA5 data. A Jupyter notebook would be sufficient as a way to start. However, dealing with different stations leads to different preprocessed files for numerous evaluation and training processes, and the situation becomes increasingly intricate. Management is needed to allow for pipelines to be run in a structured way, ensuring that the input settings for CRAI are set dynamically and that all the necessary files are created and stored in the right place. Thus, it appears natural to implement a set of functions and structure the process through an object-oriented approach. As seen in Figure 6.1 and Figure 6.2, pipelines for training and infilling have many similarities, and a lot of the implemented methods can be reused. Moreover, the model's evaluation is conducted similarly for infilling and validation. The only distinction between these two pipelines is in the final step of displaying the results." This chapter is about laying the foundation through the implementation of the different steps of the pipeline as classes and functions. Thereupon, in chapter section 7, the next abstraction level of the software into classes that handle the execution of the different steps is described.

The following subsections highlight the critical steps of the pipeline and how their func-

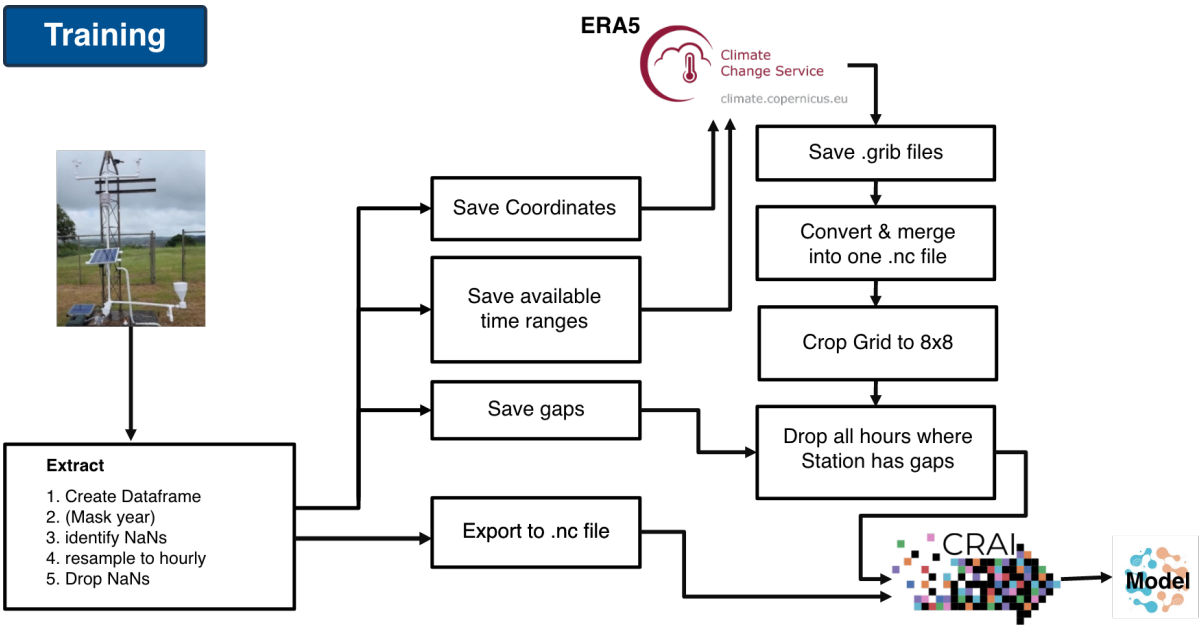


Figure 6.1: Pipeline to train a model

tionality evolves. It does not cover the detailed embedding of the methods in their classes, nor is it explicitly mentioned where temporary folders would be created to establish a philosophy where each step of the pipeline is implemented independently and can be connected in a higher-level class. The methods are not necessarily intended to be used in any arbitrary order but rather in accordance with the desired pipelines. However, the modularity allows for a better understanding and cleaner code, which is essential when developing another layer of abstraction later.

6.2 Station Data Submission and Conversion

The station data for the 3D-printed weather stations provided by NCAR comes in delimited text files, with one file per day and a text-based metadata file holding information such as the station name, latitude, longitude, and elevation. The data files (.dat files) contain one column per sensor and one row per minute.

A class `DatToNcConverter` is implemented with the following functionalities: First, it takes the `.dat` files and the metadata file into a directory and parses them. Second, the data is processed, including dropping missing values, resampling to an hourly frequency, and converting from Celsius to Kelvin to match ERA5. This is easiest in a pandas dataframe. Third, it converts the data to a `NetCDF` file, which is the format used by the ERA5 data and the CRAI module. The `NetCDF` file is structured in a way that stores the data in a 3D array with the dimensions time, latitude, and longitude. The metadata is

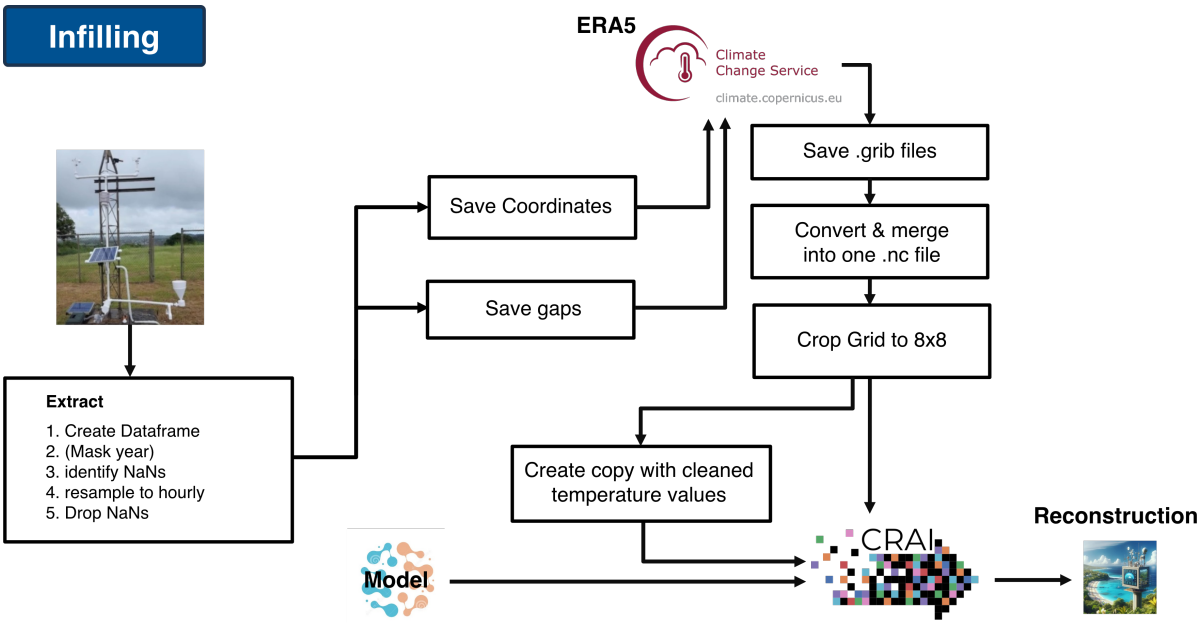


Figure 6.2: Pipeline to reconstruct weather data using a model

stored as global attributes in the `NetCDF` file. Fourth, it converts some dataframes back to `.dat` files after the reconstruction of measurements to match the original format when infilling.

The resampling to hourly frequency is the most complex use case, as it includes many design decisions. Missing values are marked with `-999.99`, so in the first step, these values are marked as `NaN`. However, the data quality is not controlled by default, and there could be values that should be marked as missing but are not. Thus, the NCAR consulted to mark `0.00` °C as `NaN`. For stations like Barbados, this wouldn't be a realistic value, but even for regions where 0°C degrees are reached often, it is unlikely to measure exactly `0.00`, meaning the amount of correct data lost through marking `0.00` as `NaN` is limited. Additionally, by agreement with the NCAR, everything above or under `± 45°C` is marked as `NaN`. To compensate for peaks in the data, aggregating the minutes using a median instead of a mean is a good idea; `numpy.median()` would return `NaN` if any value is `NaN`, while `numpy.nanmedian()` would ignore `NaN`s. It is best to have a custom aggregate function using `numpy.nanmedian()`.

For the use case of converting the data back to a `.dat` file, the dataframe is stored directly in the converter object as the original dataframe. Only the detection of `NaN`s and the aggregation to hourly values are done before, as the use case lies in the infilling task, and the reconstruction gives hourly values. After saving the original state, any transformation of units (Celsius to Kelvin), renaming of columns and most importantly, the actual clearing

of the NaNs take place. Fundamentally, in between the first and last measurements, rows exist for all minutes, even if measurements are missing. However, if the station had severe issues, it is possible that between the first and last record, there are even rows or files missing. To assure that the original dataframe, where the NaNs are not supposed to be cleared, will have rows for all hours, a handy method provided by the Python pandas module is used:

```
    self.original_df = self.original_df.reindex(
        pd.date_range(start = self.dataframe.index.min(), end = self.↩
            dataframe.index.max(), freq = "h"))
```

A class Station is implemented to hold the metadata and the pandas dataframe of the station data. It includes the converter itself, which it can then use automatically. The class primarily manages access to the different files and the data, both before and after the conversion. Additionally, it detects gaps in the data, providing a simple list of hours where data is missing for infilling purposes. For training use cases, it also lists the months where at least some data is available, which is convenient when using the API to obtain ERA5 data for the station.

```
def find_gaps(self) -> None:
available_hour_steps = self.df.index
all_hour_steps = self.converter.original_df.index
# find all hours between the first and last hour that are missing
missing_hours = all_hour_steps.difference(available_hour_steps)
return missing_hours.tolist()
```

Code Snippet 1: Gap Detection in Station Class

```
def get_all_months_in_df(self) -> None:
# return all (year, month) tuples in the dataframe
periods = self.df.index.to_period('M').unique().tolist()
month_dict = {}
for period in periods:
    if period.year not in month_dict:
        month_dict[period.year] = []
    month_dict[period.year].append(period.month)
return month_dict
```

Code Snippet 2: Detection of available ranges in Station Class

6.3 Copernicus Climate Data Store - CDS API

The Copernicus Climate Data Store (CDS) is a service by the European Centre for Medium-Range Weather Forecasts (ECMWF). The CDS API provides open source access to the ERA5 data, allowing users to download the data for a specific location and time period. After creating an account online an API Key can be obtained for free and with the python module 'cdsapi' data can be downloaded then easily in `.grib` format.

```
import cdsapi

class Era5DownloadHook:

    def __init__(self, lat, lon):
        self.cds = cdsapi.Client(
            url="https://cds.climate.copernicus.eu/api/v2",
            key=f"{os.getenv('UID')}:{os.getenv('API_KEY')}"
        )
        self.lon, self.lat = lon, lat

    def _download(cds, date_info, save_to_file_path):
        cds.retrieve(
            'reanalysis-era5-single-levels',
            {
                "product_type": "reanalysis",
                "format": "grib",
                "variable": "2m_temperature",
                "area": [
                    self.lat + 1, # limit north
                    self.lon - 1 % 360, # limit west
                    self.lat - 1, # limit south
                    self.lon + 1 % 360, # limit east
                ],
                "year": date_info.get("years"),
                "month": [f"{month:02d}" for month in date_info.get("months")],
                "day": [f"{day:02d}" for day in date_info.get("days")],
                "time": [f"{hour:02d}:00" for hour in date_info.get("hours")]

```

```
30            }, save_to_file_path)
```

Code Snippet 3: Download Hook for ERA5 Data

As can be seen in the Code Snippet 3, the request encompasses, in addition to dynamic spatial cropping and temporal selection, the fundamental details of our requirement: to download 2-meter temperature data in `.grib` format from the ERA5 reanalysis. In the class `Era5DownloadHook`, further the following is implemented to download a few hours in a day at once.

```
1     def download_hours_in_same_day(self, year, month, day, hours, ↩
          target_folder):
2         self._download({
3             "years": [year],
4             "months": [month],
5             "days": [day],
6             "hours": hours
7         }, f"{year}_{month}_{day}.grib")
```

Code Snippet 4: Download Hours in Same Day in Download Hook Class

When using a variation of Code Snippet 4to download a month or an entire year, it becomes obvious why the Code Snippet 2 is so helpful.
In the Code Snippet 3, it can be seen that the regional selection is always +/- 1 degree around the Station location. Because of the grid interval of 0.25°, this ensures that the downloaded area always includes at least 9x9 grid points, such that when cropping to an 8x8 selection the station can always be centered, even without exact knowledge of the coordinates of the desired grid points prior to requesting.

6.4 Data Preprocessing

As pointed out in subsection 6.3, the data is downloaded in `.grib` format. Using the program `cdo`, the data can be quickly converted to `.nc` format with the following simple command:

```
1 cdo -f nc copy {source_path} {nc_path}
```

However, to have the "temperature at surface" variable name unified as `"tas"` in the `NetCDF` file, the following command can be used:

```
1     import xarray as xr
2     import subprocess
```

```
3
4       def _rename_variable(self, var_name, tas_name, input, output):
5           rename_variable_command = \
6               f"cdo chname,{var_name},{tas_name} {input_path} {output_path}"
7           subprocess.run(rename_variable_command, shell=True)
8
9       ds = xr.open_dataset(input_path)
10      if 'var167' in ds.variables:
11          _rename_variable('var167', 'tas', input_path, output_path)
12      elif '2t' in ds.variables:
13          _rename_variable('2t', 'tas', input_path, output_path)
```

Code Snippet 5: Renaming Variable in `NetCDF` File

Then the files are merged using

```
1       cdo cat {temp_dir_path}/*.nc {era5_target_file_path}
```

before the data is cropped to the 8x8 grid around the station location, as described in 3.2. For training the data, it needs to be assured that the ERA5 data does not include timesteps that the nc file of the station data does not include so that the time dimensions are identical. This is done in two steps: First, the ERA5 data is cropped using a start-to-end date approach using the first and last date of the station data. Then, all hours that were missing in the station dataset, identified by the find gaps method in Code Snippet 1, are removed from the ERA5 data. This is done using the python module `xarray` and deleting the timesteps in batches of up to 1000 timesteps, as deleting all at once could cause issues.

The station data should have the same dimensions as the ERA5 data, so a method is applied that copies the ERA5 dataset and replaces all the temperature values with the measured value from the corresponding hour in the station data.

For evaluating the model, the same preparation of ERA5 data is done in a validation procedure. However, instead of passing the station data as expected output to the model, the file will not be filled with the measured values but with `NaN`s.

When evaluating the model not for validation but to infill missing values, the ERA5 data of course, needs to be prepared differently of course. Instead of requesting monthly data from the `cdsapi`, it is requesting only the missing hours day by day, such that the time-dimension is directly as desired, and preprocessing only needs to handle the conversion and geographical cropping.

6.5 Handling the Model Output

Because CRAI is using the encode, decode architecture as described in subsection 4.1, the output file in `NetCDF` format that the model produces includes not only one temperature value per timestep but has the same dimensions as the input ERA5 file meaning it uses the same 8x8 grid with 64 temperature values. However, since it was trained on input files where the ground truth was also laid on the 8x8 grid, the 64 temperature values in the output data of the model tend to be highly similar. To make this transformation but primarily for practical reasons, the output file will be converted to a pandas dataframe again, where the calculated mean of the 64 values is stored. Two different methods were implemented, creating different dataframes, as the model was evaluated in two different use cases. One is the validation of the model, where the output is compared to the ground truth, and as in section 5 to ERA5 nearest grid point. The other is the infilling of missing values, where the output is not compared but the original dataframe of the station data is updated with reconstructed values. Both methods take the output file of the model as input, while the validation method also needs to take the groud truth and ERA5 into account. For the ground truth, the file created by the `DatToNcConverter` is used, while for the ERA5 data the file that has been passed to the model is used. The ground truth file still uses the coordinates of the station and is not stretched to the 8x8 grid, as was done during the training. Therefore the method to create the validation dataframe `era5_vs_reconstructed_comparision_to_df` can obtain the station coordinates from that file and then decide which grid point in the ERA5 data is the nearest to the station. To highlight one last core method, `plot_n_steps_of_df` is very useful during validation and powered the standardized plotting of the results (subsection 5.4). It takes in the dataframe returned by `era5_vs_reconstructed_comparision_to_df`, and plots the temperature values of the station, the ERA5 nearest grid point, and the reconstructed values for a specific number of timesteps. One handy feature is that if a number of timesteps is given, it will plot only that amount of timesteps, allowing for a more detailed view of the results. It chooses the extract of `n` consecutive timesteps randomly, leading to the weekly extract plots in subsection 5.4. In all the created plots in subsection 5.4, the RMSE and the correlation coefficient are calculated and displayed in the title of the plot, that has also been done in the method `plot_n_steps_of_df`. Lastly, the method is also able to plot the temperature differentials instead of the absolute values when told to do so by an optional boolean flag. Then, of course after the calculation of the RMSE and correlation coefficient, it offsets the values In all these functionalities, the plotting method heavily benefits from the creation of the dataframe, as it can slice along the time dimension conveniently.

7 Process Orchestration with API and Web interface

As foretold in subsection 6.1, the idea is to create a system that allows for efficient execution of the three different pipelines: Training, Infilling, and Validation. With the foundation of the previous chapter we have all the necessary preprocessing and postprocessing routines in place. When we succeed in creating the next abstraction layer, allowing for easy execution of the different pipelines, such that they handle their process autonomously, we can present an end-to-end solution.
The achieved end-to-end solution comes with a web interface, allowing for an easy user experience. It gives the user everything they need to have control over the entire process, from submitting a dataset to downloading validation/training or infilling results. Such a web interface requires defined touchpoints between the user's web browser and the server where the system is running. A solution featuring programmatically defined touchpoints is called an API (Application Programming Interface), where a defined endpoint is available for each use case. An overview of the API endpoints is given in subsection 7.3.

7.1 Executor Classes: Training, Infilling, Validation

The challenge for the foretold web interface and API solution is to store different station data independently and to control processes for each station dynamically. This begins with passing the corresponding Station object to the executor class, which needs to initialize the process, including the creation of dedicated temporary folders. The executors are designed as classes to allow for easy access to the different data of the process entity itself at all steps, from ERA5 download to plotting. For example, it ensures that when plotting after the CRAI module has finished the evaluation.
The `TrainingExecutor` class is structured such that all requisite inputs are directly passed during object creation. This is mainly the station object (see section 6) which takes care of the data extraction. However, the following pipeline steps are not triggered from the class constructor automatically but from an additional method named `execute()`. This design choice enables the execution of a similar `execute_with_sbatch()` version, particularly useful when utilizing the code stack on a supercomputer like Levante at the German Climate Computing Center. The execution initially handles ERA5 data download, a process encompassing several routines detailed in section 6, such as identifying available timesteps, downloading data, merging `.grib` files, converting to `NetCDF` files, and cropping dimensions as described earlier. The robustness of the work in the previous chapter becomes evident here, with all processes controlled effortlessly and unaffected by

temporary folder management intricacies. Following the completion of the preprocessing steps, when coherent training files are stored in the executor object's temporary folders, it generates the `training-args.txt`. This file serves as input parameters for the separate machine learning code stack, specifying training execution details and output locations, thereby ensuring hassle-free model saving.
The `EvaluationExecutor`, utilized for the infilling routine, mirrors the structure of the `TrainingExecutor`, with the distinction that input parameters must include a model path. Moreover, the download routines in the `EvaluationExecutor` are adjusted to obtain ERA5 data for the hours where measurements are missing, rather than for the hours where measurements are available. The class includes an `execute()` function as well, sequentially running the steps of the pipeline (see Figure 6.2). The `training-args.txt` file for the CRAI module is generated similarly, albeit with parameters pertinent to evaluation. The output of the CRAI evaluation is then returned, such that where ever the executor is called from, the results can be processed further. Either for infilling, or as in the `ValidationExecutor` for plotting.
The `ValidationExecutor` is mainly a wrapper for the `EvaluationExecutor`. If the `EvaluationExecutor` is called without modification, it will evaluate the model over the timesteps where measurements are missing for the passed `Station`. Therefore, the `ValidationExecutor` benefits from the `EvaluationExecutor` but can't run the `EvaluationExecutor.execute()` method directly. Instead, it calls the steps of the pipeline individually. This way, it can skip the `EvaluationExecutor`'s download routine and download the ERA5 data for the hours where measurements are available.

7.2 Web interface

The web interface is the peak of the system, as it allows for easy interaction through out the full training, validation and infilling. It's various features are described in the following. The capabilities of the web interface are based on the API endpoints, which are described in subsection 7.3.
Additionally, it is possible to monitor the progress of the process through the API. The details on the API are described in subsection 7.3. This lays out a sufficient foundation for the implementation of a web interface, which is the main focus of this section.

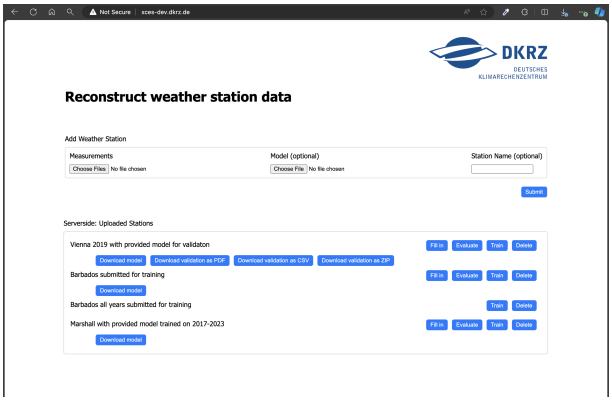


Figure 7.1: Screenshot of the webinterface

The interface consists of two areas, one where the user is supposed to submit a "dataset", which in this context is meant as a collection of measurements from a weather station and the station's information such as location, and optionally a custom name and third a trained model for the station that can be provided optionally as well. If no model is provided the system can train a model and will attach it.
The second area is a list of all datasets that the user has submitted. The API implementation allows for user identification through a unique token that is passed on to the server when the user submits a dataset, such that the user can then ask to see all datasets owned by them using that same token. The web interface automatically generates a token for the user and stores it then in the local storage of the browser, such that the user does not have to remember it.
As examples in Figure 7.1 show each dataset depending on its state has different options available. The deletion and train buttons all have in common, as no model is necessary to train a model. Datasets where a model has been provided still could be used to train a new model overwriting the old one. The `Evaluate` and `Fill in` buttons to evaluate the model over the timesteps where measurements are available, respectively to evaluate the model over the timesteps where measurements are missing.
Once a process such as training, evaluation, or infilling has been completed for a dataset the user can download the results anytime through the new buttons that appear in the dataset list. For example, it can be seen in Figure 7.1 that for the Vienna example, validation has been completed, and the user can now download a PDF with all the plots that compare the predictions with the actual measurements, or a `CSV` file with the same data, or a zip file that contains both and the images used in the PDF. The first Barbados example has a model attached that can be downloaded, meaning either it was provided or generated through training on the server itself. The second Barbados example has no model attached yet, meaning only the buttons "Train" and "Delete" are available because no model was provided and no training has been done yet.

7.3 API Endpoints

As introduced above, the Application Programming Interface (API) is a set of Hypertext Transfer Protocol (HTTP) request messages that define the communication between the web interface and the server. When a user interacts with the web interface, such as pressing a button, it typically triggers an action on the server or retrieves information.

The web interface accomplishes this by sending the appropriate HTTP request to the server. The server processes the request and responds with the necessary information, which is then displayed on the web interface.
Below is a brief description of the available endpoints. The notation `<user-token>` and `<dataset-id>` are placeholders for the actual user token and dataset ID to identify the correct user or dataset. The user can choose a unique user token for themself; the web interface does that automatically and stores it in the user's browser, while the dataset ID is generated by the server when a dataset is submitted.

POST /data-submission

This endpoint is used to pass a list of measurement files (`.dat`) and a metadata file (`.rtf`) to the server. Additionally a model file (`.pth`) Each dataset is associated with a unique ID, and a unique token representing the owner. The owner's identification token needs to be passed in the request body. Also, a custom name for the dataset can be passed with the request body. The dataset ID is created by the server and returned in the response. The dataset ID is used to refer to the dataset in the following API calls.

GET /available-datasets/<user-token>

For a given user token, this endpoint returns a list of all datasets that the user has submitted. The user token is passed as a URL parameter. The response is a list of dictionaries, each containing information about the dataset, such as the dataset ID, the name, and the status of the dataset (e.g. if it is busy training a model and progress percentage). Moreover, for each dataset, a boolean flag indicates whether a trained model, a validation result or an infilling result is attached. If so, the user can download the results (the last three endpoints).

GET /train/<dataset-id>

A training process is initiated by passing the `Station` object of the corresponding data submission to the `TrainingExecutor`. When the training process is finished, the model is saved in the data submission object, and if not timed out yet, the request is answered with the model. The model is then available through the respective download endpoint.

GET /validate-model/<dataset-id>

Starts the model validation process for the specified dataset. The process includes downloading ERA5 data for times with available measurements and comparing the model predictions with actual data. Validation results, including a PDF and CSV file, are saved in the dataset object and can be downloaded using the respective endpoints.

GET /fill-in/<dataset-id>

Initiates the infilling process for the specified dataset. The process involves using the model to predict missing measurements and generating results that include both the infilled data and related plots. The infilling results are saved in the dataset object and can be downloaded using the respective endpoint.

DELETE /delete-dataset/<dataset-id>

Deletes the specified dataset from the server along with any associated models, validation results, and infilling results. The server responds with a confirmation message upon successful deletion.

GET /download-model/<dataset-id>

Allows the requestor to download the trained model associated with the specified dataset. The server responds by sending the model file as an attachment.

GET /download-validation-zip/<dataset-id>

Allows the requestor to download the validation results as a ZIP file containing both the PDF and CSV files, along with related images. The server responds by sending the ZIP file as an attachment. There are also endpoints to download the PDF and CSV files exclusively.

GET /download-infilling/<dataset-id>

Allows the requestor to download the infilling results associated with the specified dataset. The server sends the infilling results file as an attachment if available.

8 Conclusion

This thesis employed a machine-learning technique to reconstruct temperature data at weather stations, offering a computationally lightweight alternative to Numerical Weather Prediction (NWP). The approach was specifically designed to utilize only the ERA5 reanalysis data as input, making it universally applicable to any weather station worldwide. Trained models for each station, based on convolution layers, effectively identified local patterns in the ERA5 data, resulting in more accurate temperature predictions compared to the ERA5 data itself. Upon validation against test data, the models consistently outperformed the ERA5 data, showcasing varying levels of accuracy across different datasets. Despite limitations in training data availability, the models successfully corrected both major and minor biases present in the ERA5 data. However, the correlation coefficient wise performance was more closely tied to the quantity of training data. Notably, the model trained for the weather station in Barbados displayed promising capabilities in reconstructing the diurnal cycle, a feature not adequately represented in the ERA5 data despite being trained on only two years of data.

Looking forward to future research, various extensions for improvement and application can be explored. One potential area is the incorporation of additional climate variables. The weather stations of the 3D-PAWS network, for example, are equipped with a solar-radiation sensor. If only the temperature data is missing, the model could be trained to take besides the ERA5 temperature data, the solar radiation data as input.

Although reconstruction is the pillar of this research, the ultimate goal is to divulge outstanding results to weather station operators, such as the 3D-PAWS team. The thesis has developed an end-to-end software solution accompanied by a user-friendly web interface. This interface enables the reconstruction of temperature data at any weather station globally without requiring extensive computer science knowledge. It is crucial to note that the model's performance relies significantly on the quality and quantity of training data. Hence, the software provides a platform for users to train and validate the model with their own data, accessible not only through the web interface but also automatically via the provided API. Additionally, users can utilize the model's reconstruction results to assist in error-checking datasets by comparing them with actual measurements. Subsequently, on the cleansed dataset, the model can be retrained, leading to improved accuracy. The modularized design of the software allows for adjustments and extensions, such as incorporating different Reanalysis data sources. Suppose another reanalysis dataset is more accurate for a specific region. In that case, the software can be configured to utilize this dataset instead of ERA5, with the models subsequently retrained

on the new data. Furthermore, it can be explored how the model can be applied to existing weather forecast data, such as that from the European Centre for Medium-Range Weather Forecasts (ECMWF) or the National Oceanic and Atmospheric Administration (NOAA). This could be a promising approach to improve the local forecast quality, as the model represents knowledge of the local weather patterns.
In conclusion, the research presented in this thesis has demonstrated the potential of machine learning in reconstructing temperature data at weather stations in a computationally efficient and universally applicable manner.

References

[1] A. Ortiz-Bobea. Climate, agriculture and food, 2021. URL `https://doi.org/10.48550/arXiv.2105.12044`.

[2] M. N. Mistry, R. Schneider, P. Masselot, and et.al. Comparison of weather station and climate reanalysis data for modelling temperature-related mortality. *Scientific Reports*, 12(1):5178, 2022. doi: 10.1038/s41598-022-09049-4. URL `https://doi.org/10.1038/s41598-022-09049-4`.

[3] R. Muita, P. Kucera, S. Aura, D. Muchemi, D. Gikungu, S. Mwangi, M. Steinson, and et al. Towards increasing data availability for meteorological services: Intercomparison of meteorological data from a synoptic weather station and two automatic weather stations in kenya. *American Journal of Climate Change*, 10:300–303, 2021. doi: 10.4236/ajcc.2021.103014. URL `https://doi.org/10.4236/ajcc.2021.103014`.

[4] S.Y. Chung, S. Venkatramanan, H. E. Elzain, S. Selvam, and M. V. Prasanna. Chapter 4 - supplement of missing data in groundwater-level variations of peak type using geostatistical methods. In *GIS and Geostatistical Techniques for Groundwater Science*, pages 33–41. Elsevier, 2019. ISBN 978-0-12-815413-7. doi: https://doi.org/10.1016/B978-0-12-815413-7.00004-3.

[5] William C Skamarock, Joseph B Klemp, Jimy Dudhia, David O Gill, and et al. A description of the advanced research wrf version 3. *NCAR technical note*, 475:113, 2008.

[6] T. Kurth, S. Subramanian, P. Harrington, J. Pathak, M. Mardani, D. Hall, and et al. Fourcastnet: Accelerating global high-resolution weather forecasting using adaptive

fourier neural operators. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, PASC '23. Association for Computing Machinery, 2023. doi: 10.1145/3592979.3593412. URL https://doi.org/10.1145/3592979.3593412.

[7] K. Bi, L. Xie, H. Zhang, X. Chen, X. Gu, and Q. Tian. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 619(7970):533–538, 2023. doi: 10.1038/s41586-023-06185-3. URL https://doi.org/10.1038/s41586-023-06185-3.

[8] R. Lam, A. Sanchez-Gonzalez, M. Willson, and et. al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023. doi: 10.1126/science.adi2336. URL https://www.science.org/doi/abs/10.1126/science.adi2336.

[9] European Centre for Medium-Range Weather Forecasts. ERA5: Fifth generation of ECMWF atmospheric reanalyses of the global climate, 2024. URL https://www.ecmwf.int/en/forecasts/dataset/ecmwf-reanalysis-v5. Accessed: 2024-05-29.

[10] Charles Mwangi, Martin Steinson, and Paul Kucera. Low cost weather stations for developing countries (kenya). https://www.un-spider.org/sites/default/files/21.%20UNSPIDER%20Presentation%20-%20Mwangi.pdf, 2017. 7th UNSPIDER Conference, 23rd-25th October 2017. Prepared with support of: Martin Steinson/Paul Kucera - UCAR/NCAR.

[11] University Corporation for Atmospheric Research (UCAR). *3D-PAWS Manual*, 2024. URL https://sites.google.com/ucar.edu/3dpaws.

[12] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9(4):611–629, 2018.

[13] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[14] E. A. Barnes, J. W. Hurrell, I. Ebert-Uphoff, C. Anderson, and D. Anderson. Viewing forced climate patterns through an ai lens. *Geophysical Research Letters*, 46:13389–13398, 2019.

[15] E. Racah and et al. Extremeweather: A large-scale climate dataset for semi-supervised detection, localization, and understanding of extreme weather events. In *Advances in Neural Information Processing Systems*, volume 30, pages 3405–3416, 2017.

[16] C. Kadow, D. M. Hall, and U. Ulbrich. Artificial intelligence reconstructs missing climate information. *Nature Geoscience*, 13(6):408–413, 2020. doi: 10.1038/s41561-020-0582-5. URL https://doi.org/10.1038/s41561-020-0582-5.

[17] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer International Publishing, 2015. doi: 10.1007/978-3-319-24574-4_28. URL https://link.springer.com/chapter/10.1007/978-3-319-24574-4_28.

[18] G. Liu, F. Reda, K. Shih, T. C. Wang, A. Tao, and B. Catanzaro. Image inpainting for irregular holes using partial convolutions. 04 2018.

[19] Datahacker.rs. Edge detection. https://datahacker.rs/edge-detection/, 2021. Accessed: 2024-05-19.

[20] P. Poli, H. Hersbach, Dick P. Dee, P. Berrisford, A. J. Simmons, F. Vitart, and et. al. Era-20c: An atmospheric reanalysis of the twentieth century. *Journal of Climate*, 29(11):4083–4097, 2016. doi: 10.1175/JCLI-D-15-0556.1. URL https://journals.ametsoc.org/view/journals/clim/29/11/jcli-d-15-0556.1.xml.

[21] European Centre for Medium-Range Weather Forecasts (ECMWF). Fact sheet: Earth system data assimilation, 2020. URL https://www.ecmwf.int/en/about/media-centre/focus/2020/fact-sheet-earth-system-data-assimilation. Accessed: 2024-06-03.

[22] H. Hersbach, B. Bell, P. Berrisford, S. Hirahara, and et al. The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020. doi: 10.1002/qj.3803. URL https://doi.org/10.1002/qj.3803.

[23] R. Gelaro, W. McCarty, M. J. Suárez, R. Todling, A. Molod, and et al. The modern-era retrospective analysis for research and applications, version 2 (merra-2). *Journal of Climate*, 30(14):5419–5454, 2017. doi: 10.1175/JCLI-D-16-0758.1. URL https://doi.org/10.1175/JCLI-D-16-0758.1.

[24] S. Kobayashi, Y. Ota, Y. Harada, A. Ebita, M. Moriya, and et al. The jra-55 reanalysis: General specifications and basic characteristics. *Journal of the Meteorological Society of Japan*, 93(1):5–48, 2015. doi: 10.2151/jmsj.2015-001. URL https://doi.org/10.2151/jmsj.2015-001.

[25] S. Saha, S. Moorthi, X. R. Wu, J. Wang, S. Nadiga, and et al. The ncep climate forecast system version 2. *Journal of Climate*, 27(6):2185–2208, 2014. doi: 10.1175/JCLI-D-12-00823.1. URL https://doi.org/10.1175/JCLI-D-12-00823.1.

[26] H. E. Beck, M. Pan, T. Roy, G. P. Weedon, F. Pappenberger, and et al. Daily evaluation of 26 precipitation datasets using stage-iv gauge-radar data for the conus. *Hydrology and Earth System Sciences*, 23:207–224, 2019. doi: 10.5194/hess-23-207-2019. URL https://doi.org/10.5194/hess-23-207-2019.

[27] S. Gleixner, T. Demissie, and G.T. Diro. Did era5 improve temperature and precipitation reanalysis over east africa? *Atmosphere*, 11(9):996, 2020. ISSN 2073-4433. doi: 10.3390/atmos11090996. URL https://www.mdpi.com/2073-4433/11/9/996.

[28] City of Boulder. South boulder peak. https://bouldercolorado.gov/trail/south-boulder-peak, 2024. Accessed: 2024-05-23.

[29] M.W. Williams, M. Losleben, N. Caine, and D. Greenland. Changes in climate and hydrochemical responses in a high-elevation catchment in the rocky mountains, usa. *Limnology and Oceanography*, 41(5):939–940, 1996. doi: 10.4319/lo.1996.41.5.0939. URL https://aslopubs.onlinelibrary.wiley.com/doi/abs/10.4319/lo.1996.41.5.0939.

[30] Kelly H. Zou, Kemal Tuncali, and Stuart G. Silverman. Correlation and simple linear regression. *Radiology*, 227(3):617–628, 2003. doi: 10.1148/radiol.2273011499. URL https://doi.org/10.1148/radiol.2273011499.

Eidesstattliche Versicherung

Wacke Timo

Last Name, First Name // Name, Vorname

„Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit mit dem Titel

Wissenschaftliche Softwareentwicklung eines Convolutional Neuronal Networks für die Rekonstruktion fehlender Daten einer Wettermessstation unter Verwendung von Numerischen Modelldaten

im Bachelorstudiengang Computing in Science (Physics Specialization)selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe.

Hamburg, den June 8, 2024

Place, Date, Signature // Ort, Datum, Unterschrift