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

---

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

# Eidesstattliche Versicherung

Wacke Timo
Last Name, First Name // Name, Vorname

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem Titel

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

selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Hamburg, den May 24, 2024
Place, Date, Signature // Ort, Datum, Unterschrift

## Abstract

## Zusammenfassung

# Acknowledgements

Thank you Chris

# Contents

# List of Figures

# 1 Introduction

Weather station density varies greatly across the globe, depending on population density, economic development, and the availability of infrastructure. [2] While any weather station can experience downtime, the reliability of weather stations in regions with low station density is often low as well. So not only is downtime in regions where data is limited more likely but 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 and from a global perspective aiding climate research. For example in East Africa, the weather station density is very low, but the region would be of great interest to the El Niño Southern Oscillation (ENSO) research. The irregular fluctuation between El Niño and La Niña phases affects the climate from the tropics to even higher-latitude regions through teleconnections. [3, 4] 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. [4], either way, low-cost weather stations have reliability issues and are prone to downtime which is outlined in section 2.

In light of the challenges posed by sparse weather station coverage, novel methodologies are required to address the reconstruction of missing weather data. One promising avenue involves the application of machine learning techniques, which offer a departure from traditional numerical reconstruction methods such as kriging that are reliant on neighboring station data and are often computationally intensive. The application of machine learning in this case would be to connect numerical reanalysis data that is describes the weather in grid cells meaning it's to some degree blurry, with the local patterns that lead to measurements at a weather station. This would allow for independent operation and minimal computational resources needed for the appliance 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 endavour 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

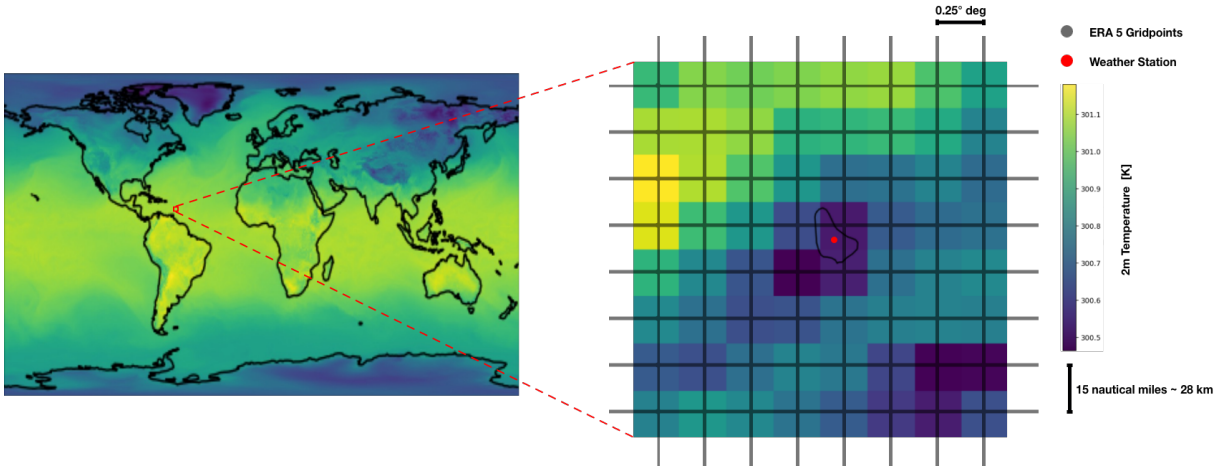



Figure 1: 8x8 grid-points of ERA5 with 2m temperature for 2020-06-23 19:00 UTC in the area of a weather station on Barbados

from 1940 to the present and contains a wide range of variables, such as temperature, precipitation, wind speed, and many more.
To prove the concept it's likely easiest to start with temperature data, meaning the 2m temperature variable from the ERA5 reanalysis will be used as input to the neural net, one hour at a time, and the expected output will be the temperature at the weather station, during the same hour.

# 2 3D printed Weather Stations

# 3 Conceptual Framework

## 3.1 Concept

The conceptual framework of the proposed method is illustrated in Figure 2. Applying the local bias of the weather station against ERA5 on top of ERA5 would reconstruct the temperature data at the station. The idea is that a Convolutional Neural Network would learn the complex behaviour of the bias depending on the arial situation, if you train it with enough available measurments. Once the CNN is trained as a model that represents the local bias of the weather station vs the ERA5 data, applying it to the ERA5 data allows for the reconstruction of missing temperature values at the weather station just by providing the data of the ERA5 temperature in the regional cutout at that timestamp used in training as an input. To illustrate what such a bias could be: In the case of Barbados for example the grid cells of the ERA5 data all lay primarly in the ocean,

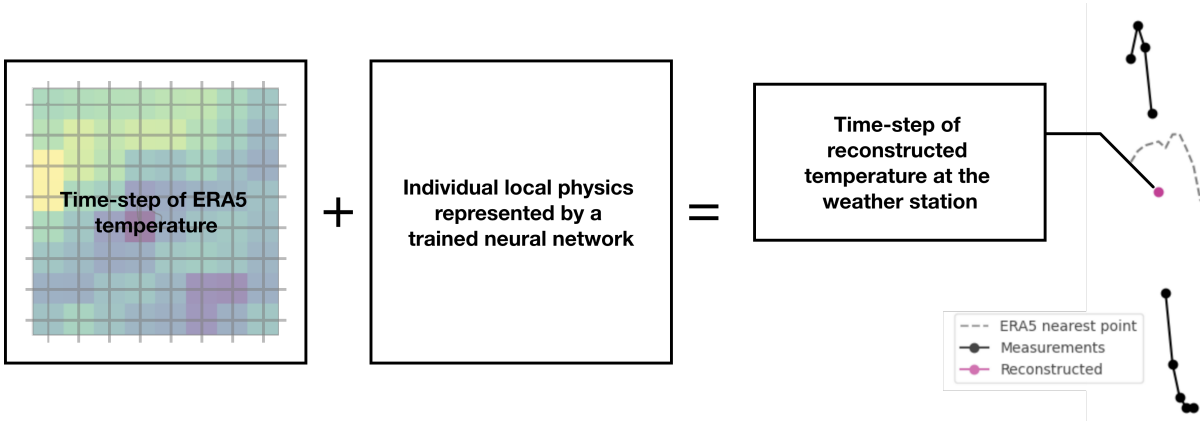



Figure 2: Conceptual Framework of the proposed method

meaning the diurnal cycle has a much lower amplitude than at the weather station that is placed on land, the neural network would need to learn how to detect based on the 64 grid points in which phase of the diurnal cycle the weather station is and then adjust the temperature values accordingly. Besides this obvious difference between ERA5 and the measurmenets at Barbados there are most likely many more local effects to master.
Since the ERA5 data is available everywhere and for any timestep, for any hour without measurements at the weather station that the model is trained for could be reconstructed. To train the model, a supvervised learning approach would be used, meaning that the input will be ERA5 data at times where measurements are available and the expected output should be the measurements at the weather station. The model would be trained to minimize the difference between the expected output and the model's prediction through backpropagation. To allow for flexibility in application and simplicity in training, the training has to be done hour by hour, meaning the model is passed during training one timestep at a time and the weights are updated between iterations. When reconstructing missed data, hence the model is applied to the ERA5 data hour by hour as well 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 coherant in time and space.
To determine after the training if and to which extent the model learned to reconstruct the missing data, the dataset pair is split again alogn the timeaxis into a pair of station with ERA5 data for training and oair 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.

## 3.3 Model Evaluation

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 timerange 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.

### 3.4 Application to New Data

Upon successful training and validation, the model is trained again with the measurements that have been exluded before in the benefit of validation. After training on the full data the CNN can be used to fill gaps. Fundamentally any list of timesteps that the model should applied for can be requested and then the ERA5 data for the respective timesteps is optained, cropped in the same way to the geographical region as before and then the model is applied to the data. 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. Since in 3.2 we split the ERA5 dataset already into two datasets, one with all hours marked as missing and one with all hours marked as present, the model can be applied to the dataset with all hours marked as missing and then directly infilled into the original measurements dataset.

## 4 Theoretical Background

### 4.1 Convolutional Neural Networks

The application of CNNs in climate science has yielded several notable contributions. According to the sources provided, CNNs have excelled at extracting patterns and features from spatial data such as satellite imagery, radar data, and climate model outputs. This capability has enabled researchers to better understand and predict complex atmospheric and oceanic phenomena. For instance, CNNs have been employed to detect and localize extreme weather events from satellite data, enhancing early warning systems and disaster response efforts. Furthermore, CNNs have improved the ability to identify forced climate patterns and disentangle natural variability from human-induced climate change signals, advancing our understanding of climate dynamics and informing mitigation strategies.
Convolutional neural networks (CNNs) are a type of deep learning architecture inspired by the visual cortex of animals. They are designed to efficiently capture spatial and temporal dependencies in data through the use of learnable filters and hierarchical feature representations. 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

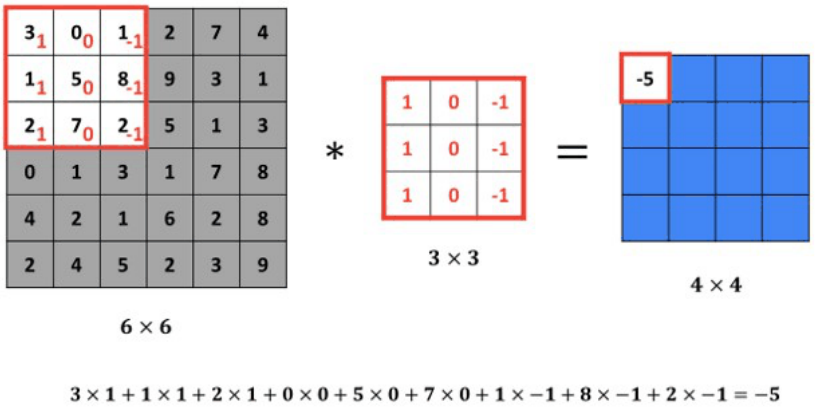



Figure 3: How a convolution operation works. [1]

necessary computational resources.
Application of CNNs in climate science has yielded several notable contributions, including the reconstruction of the El Nino event of 1877 by Kadow et al. despite extremely limited data availability. [5]

### Convolutional Layer

The convolutional layer is the driver for feature mapping in a CNN. It applies a convolution operation shown in Figure 3 to the input data. The operation is done element by element while sliding a filter (also called kernel) over the input data such that situations, where the filter overflows the input data ranges, are avoided or taken care of. 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 3 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

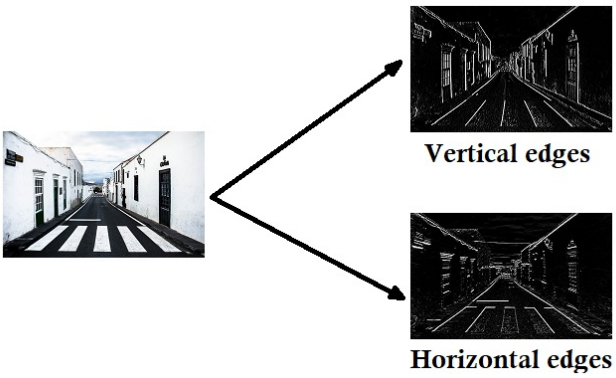


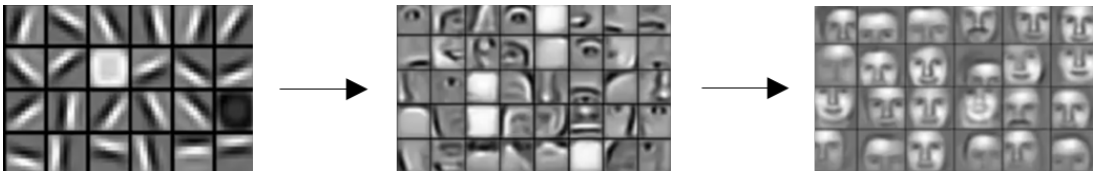

Figure 4: Example of edge detection with a convolutional kernel. [1]

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.
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.

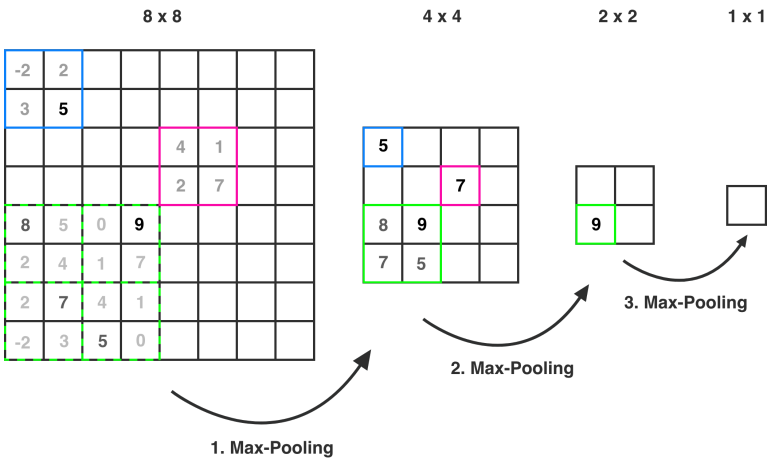
### Activation Function

### Pooling Layer

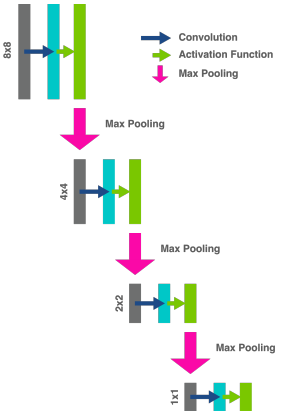
The exact position of a low-level feature in the input data is not so important when it comes to detecting high-level features, it is more important to recognize if a feature is present at all in certain spatial areas of the input data or not. Thus scaling down the resolution of the matrix by combining every 2x2 submatrix into one value can be beneficial. It is most commonly aggregated by taking the maximum value of the submatrix because it works for the mentioned purpose to detect if a feature is present in the area of the submatrix or not. While the convolution layer depending on how the convolution is processed near the borders of the input data reduces the dimensionality of the input data just slightly or not at all, the pooling reduces the dimensionality drastically. So after a 2x2 pooling operation, the output matrix is only a quarter of the size of the input matrix. For the 8x8 grid cells of the ERA5 data, that is used as input in the laid out approach (see section 3), 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 5b. Figure 5b just illustrates the downsampling of the data, in the actual architecture pooling layers always follow convolutional layers and are never applied directly after each other as seen in Figure 5c.



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




(b) Illustration of 3 max pooling layers.




(c) How Convolutional layers and pooling layers alternate

Figure 5: Abstraction in the encoding part of a CNN.

### Upscaling Layer

#### Skip Connections

#### Supervised Learning

## 4.2 Reanalysis - ERA5

# 5 Results

## 5.1 Included Weather Stations

From the 3D-Paws project of NCAR, measurements from three different weather stations in recent years were obtained. The stations are located in Marshall, Colorado; Vienna, Austria; and Barbados, Caribbean. The measurements span up to late 2023, with the quantity of available temperature data varying significantly between stations. The station in Marshall is one of the oldest in the 3D-Paws project and has the most available measurements among the three stations. The weather station in Vienna also has data dating back to 2017, but it has been almost completely offline since mid-2019. The weather

station in Barbados began recording in mid-2020
The station in Marshall is located between Boulder and Denver next to Marshall Lake, on an Elevation of 1743m, less than 10km east of the South Boulder Peak, which is already more than 2600m high [6]. That the station is located just about 30 kilometers east of the Colorado Front Range where the Rocky Mountains range in their altitude between 3250m - 4000m [7] is visible in the ERA5 data Figure 7.
Vienna is situated to the west within the Vienna Basin, at a relatively low altitude just beyond the Alpine region. The Alps, which rise moderately western of Vienna, reach heights of up to 2000 meters above sea level in a distance of approximately 100km, meaning the ERA5 resolution is high enough to differentiate the atmospheric conditions and topographical features within this region better.

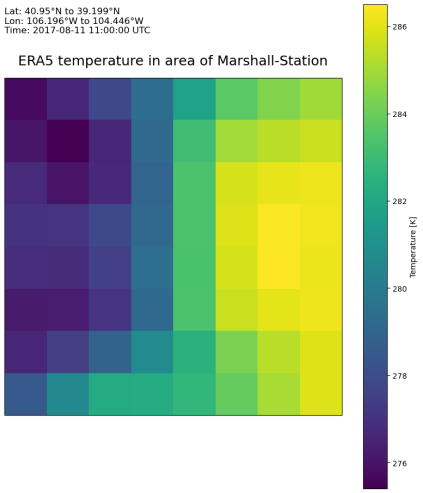
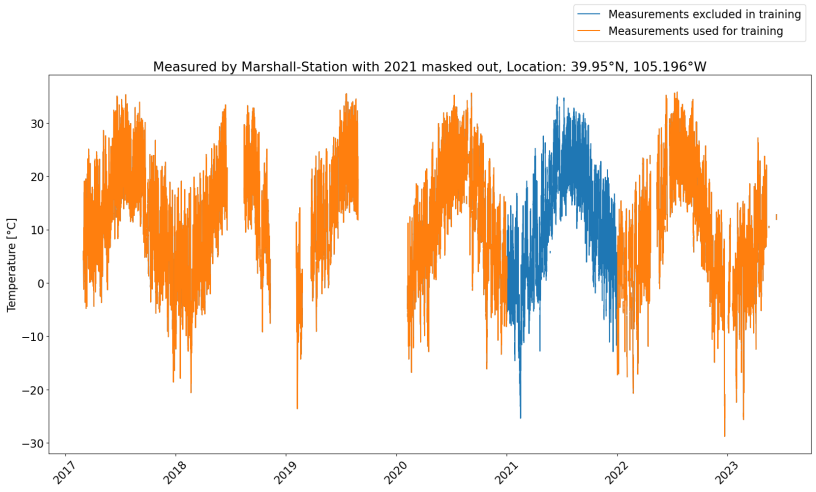



Figure 7: ERA5 Area of the station in Marshall, Colorado

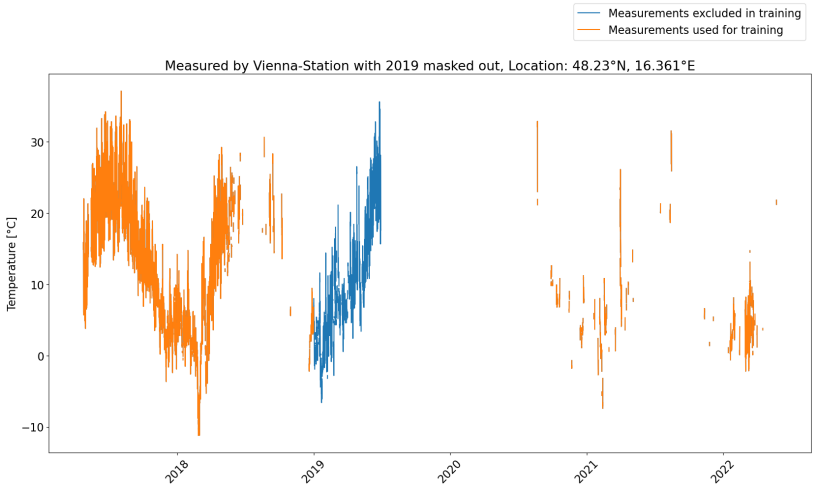
The station in Barbados is located in the parish of St George at an altitude of 274m, the island itself is relatively flat with the highest point being Mount Hillaby at 340m. The island is 34km long and up to 23km wide, which is approximately the size of an ERA5 grid cell, but as can be observed in Figure 1 all surrounding grid cells are ocean heavy, while the stations location on Barbados is close the the point where the landmass is the widest. The ERA5 data is therefore not able to capture the diurnal cycle of the station as it is located on land, while the surrounding grid cells are mostly ocean.

## 5.2 Data availability

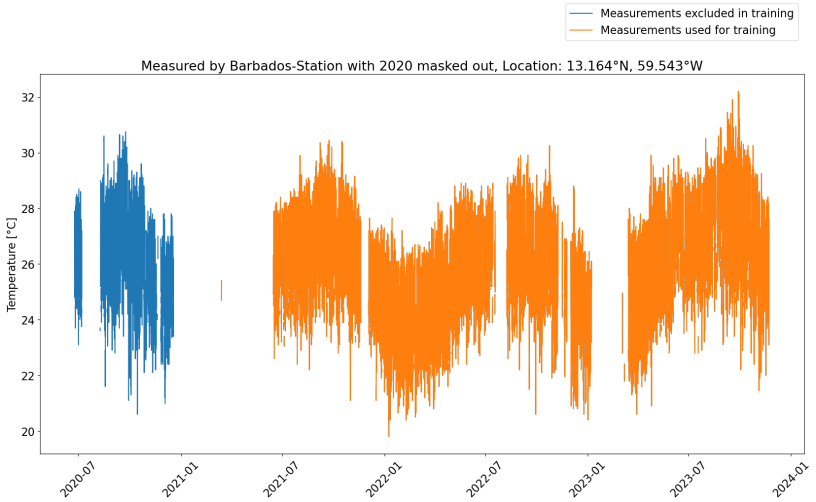
In Figure 6 the measurements of the three stations are displayed, after data cleansing and converting them on an hourly basis. The measurements were provided by the NCAR with most invalid values already marked as such but especially in Barbados the sensors had more noise and occasional invalid measurements such that they needed extra cleansing. To do so, values at temperatures near zero and below were excluded. The conversion from minute to hourly data was done by taking the average of all minute values in that specific hour, but only if there were more than 20 values available and only if the values weren't all the same. It has been observed that the sensors sometimes get stuck and then deliver the same value for a longer period. The dataset for the Marshall station spans from




(a) Station in Marshall, Colorado, USA




(b) Station in Vienna, Austria




(c) Station in Barbados, Caribbean

Figure 6: Available data from 3D printed Weather stations

2017 to 2023, comprising 41,883 data points. These measurements reveal three significant data gaps along with several smaller ones. The larger intervals without data include the midsummer period of 2018, late 2019 to early 2020, and the most extensive gap occurring over five months from September 2019 to January 2020

For the Vienna Station, measurements are available for only 12,477 hours, which is less than a third compared to Marshall. This is primarily due to extended downtimes starting from mid-2018, resulting in few to no measurement data. Except for a brief period from late 2018 to July 2019, when continuous measurements resumed. After that, there were only sporadic data collection instances, before the station went preliminary offline in April 2022.

The Barbados Station only suffered 2 downtimes that were significantly longer than a month with the biggest being roughly the first half of 2021 and the 2nd biggest being the first quarter of 2023. In total the station has 17,315 hours of measurements, which is 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 wather 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 6a 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

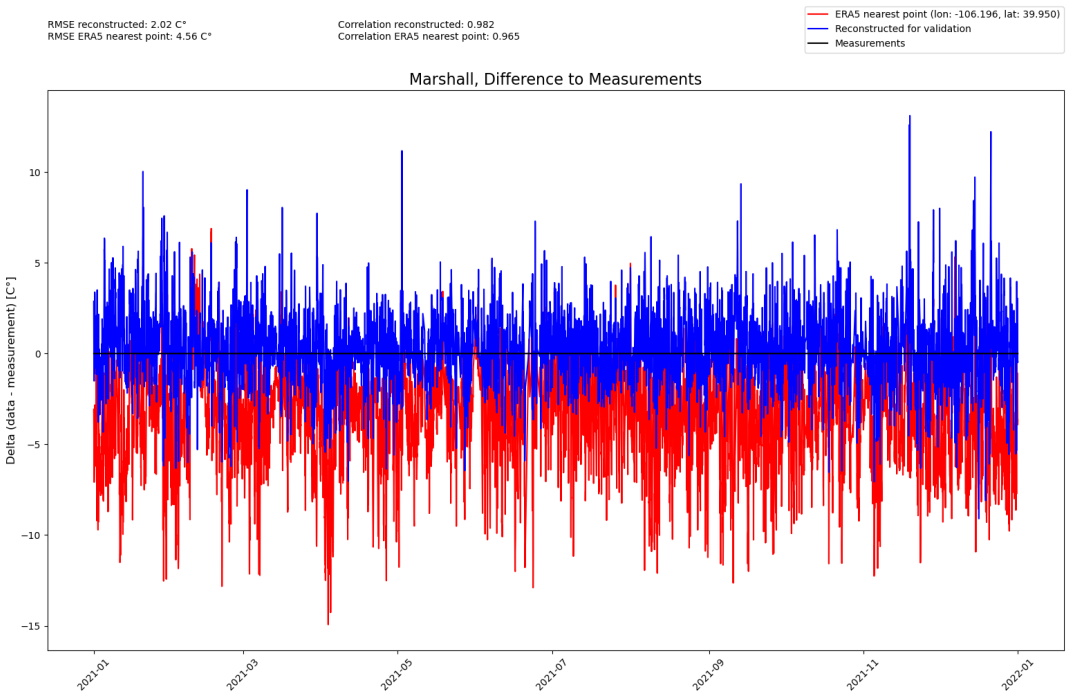



Figure 8: Difference between reconstructed and measured temperature for Marshall-Station

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.

## 5.3 Validation of the Model

### Marshall Station

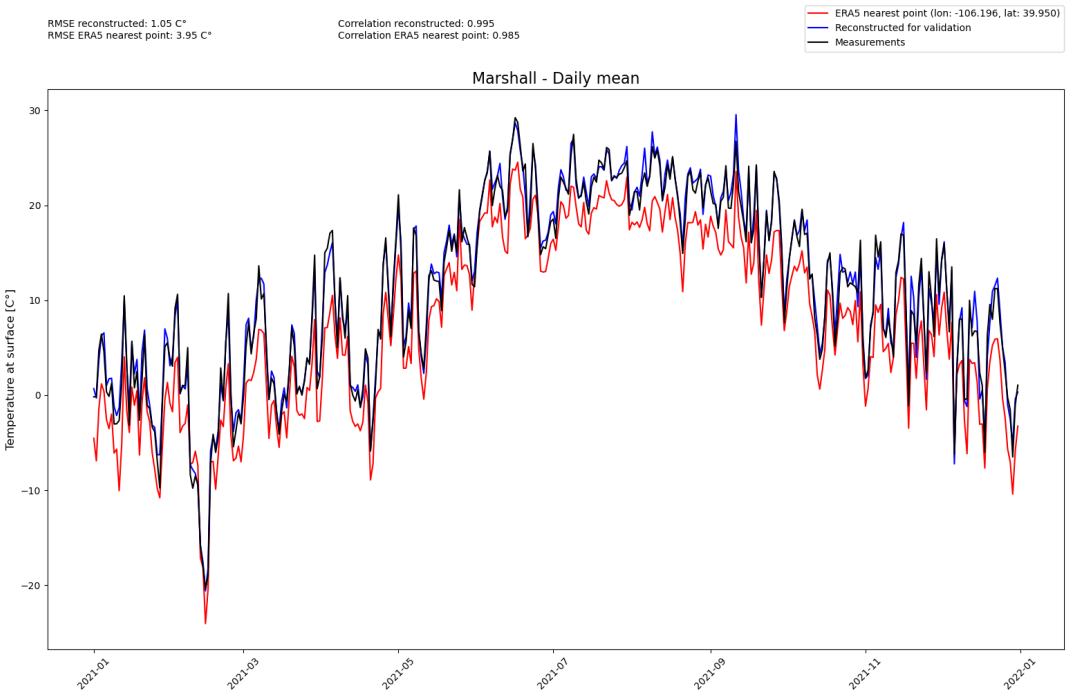



Figure 9: Reconstructed temperature vs measured temperature for Marshall-Station (Daily mean)

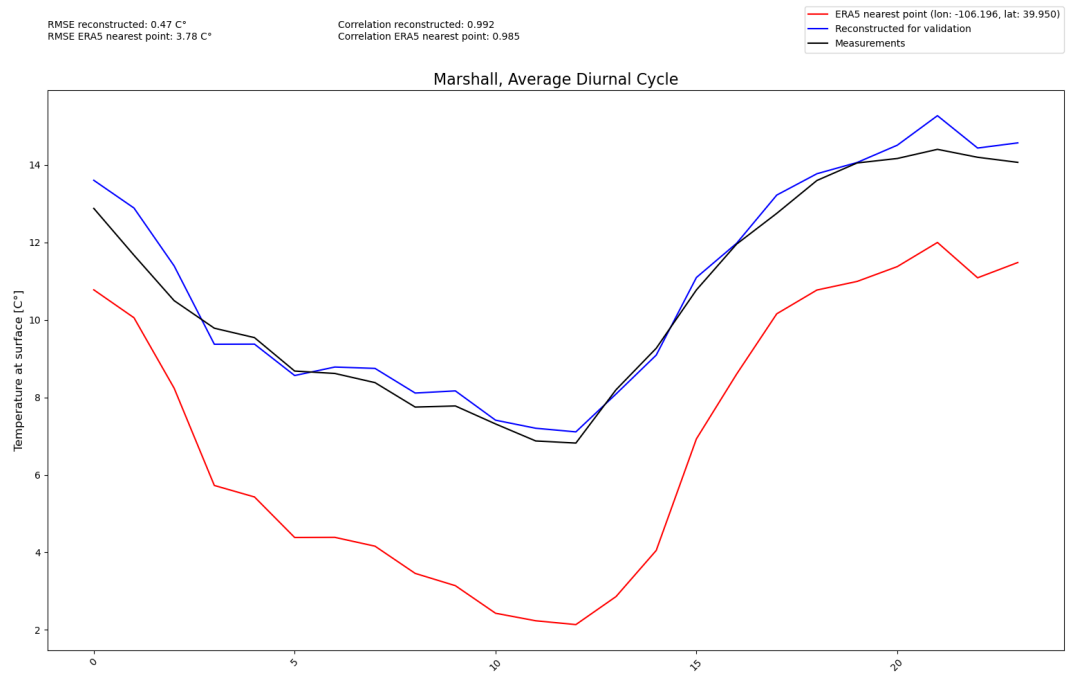



Figure 10: Reconstructed temperature vs measured temperature for Marshall-Station (Average Diurnal Cycle)

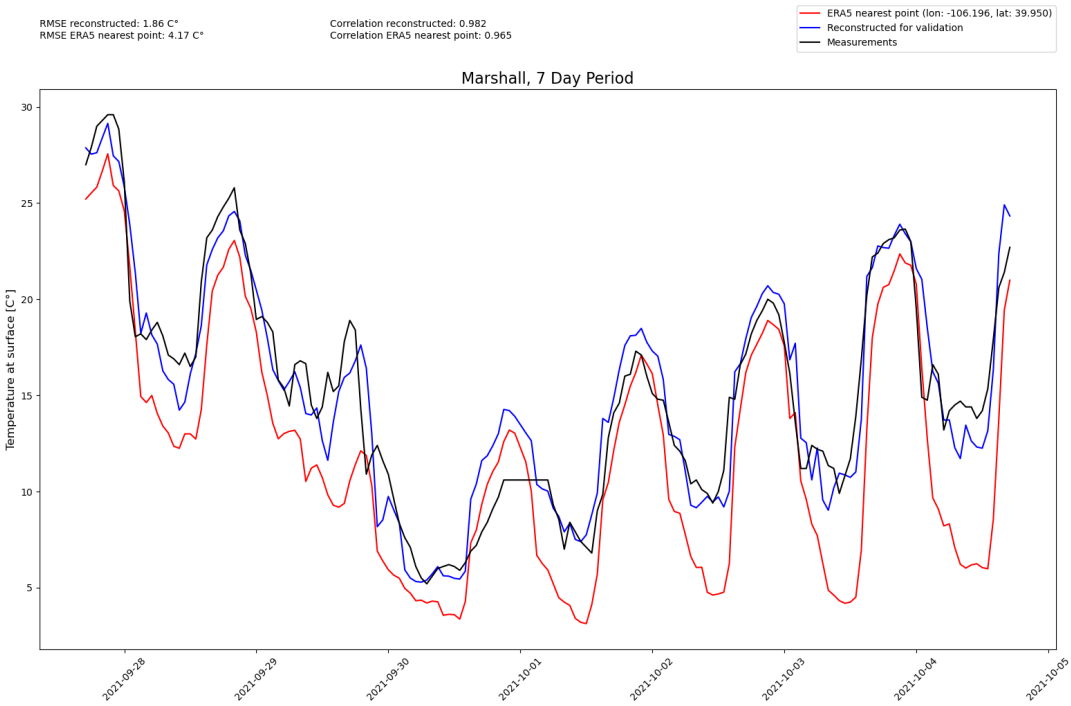



Figure 11: Reconstructed temperature vs measured temperature for Marshall-Station (7 Day Period)

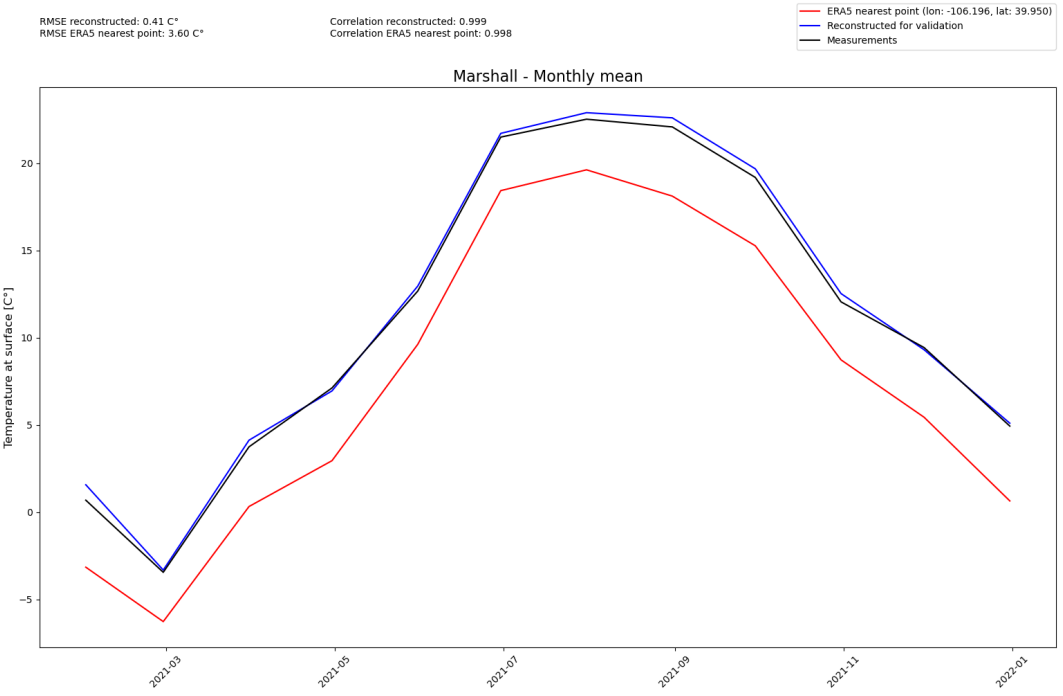



Figure 12: Reconstructed temperature vs measured temperature for Marshall-Station (Monthly mean)

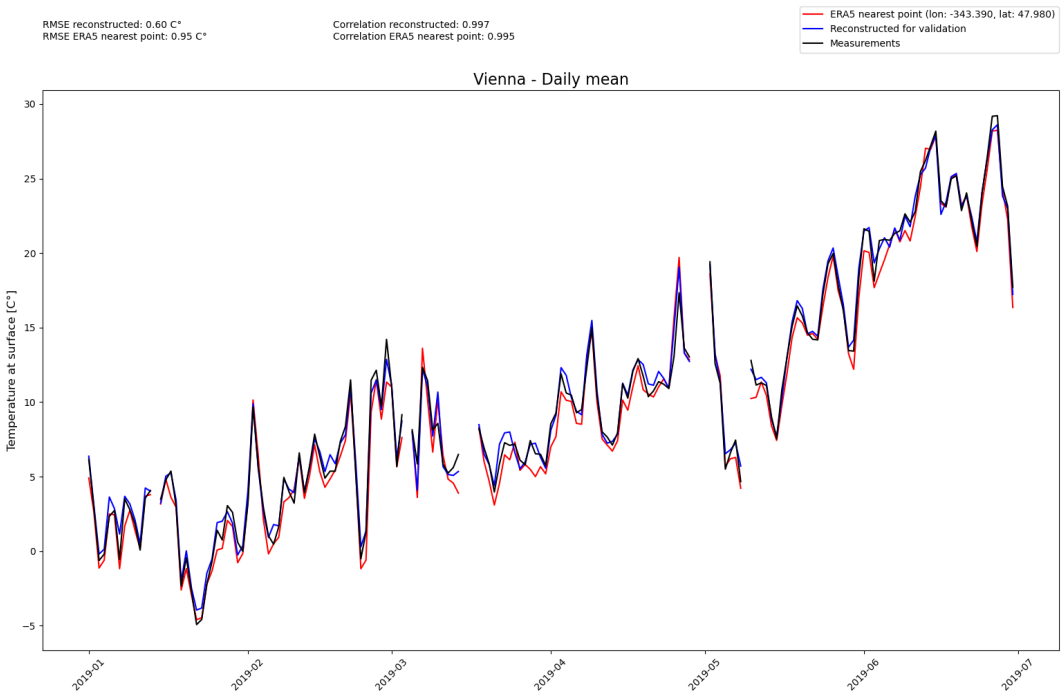



Figure 13: Reconstructed temperature for Vienna-Station (Daily mean)

## Vienna Station

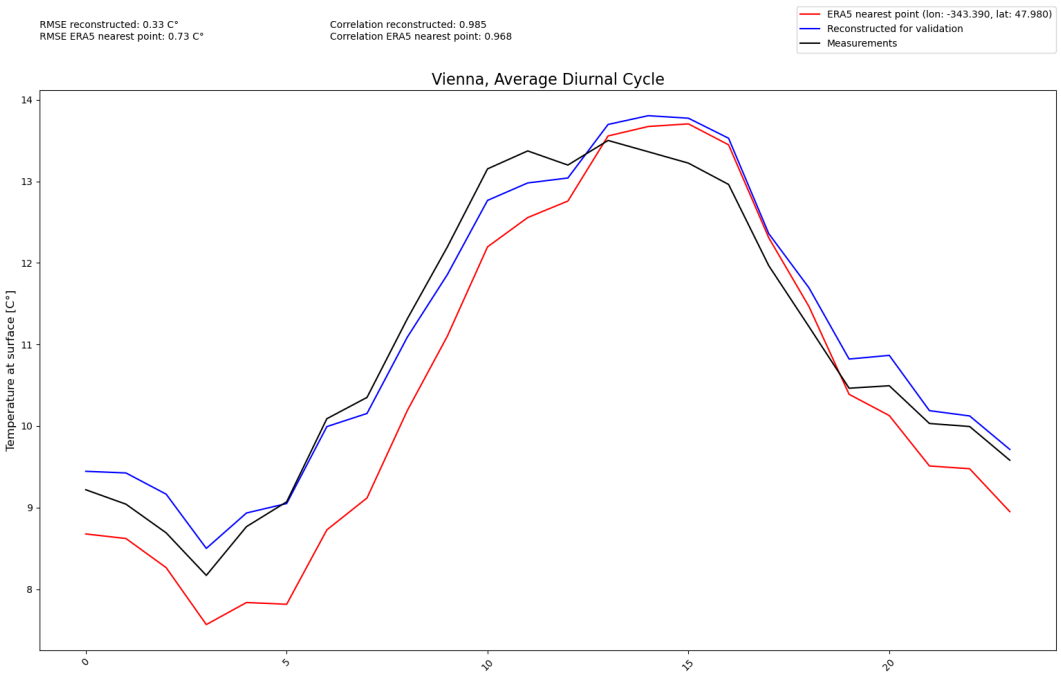



Figure 14: Measured temperature for Vienna-Station (Average Diurnal Cycle)

## Barbados Station

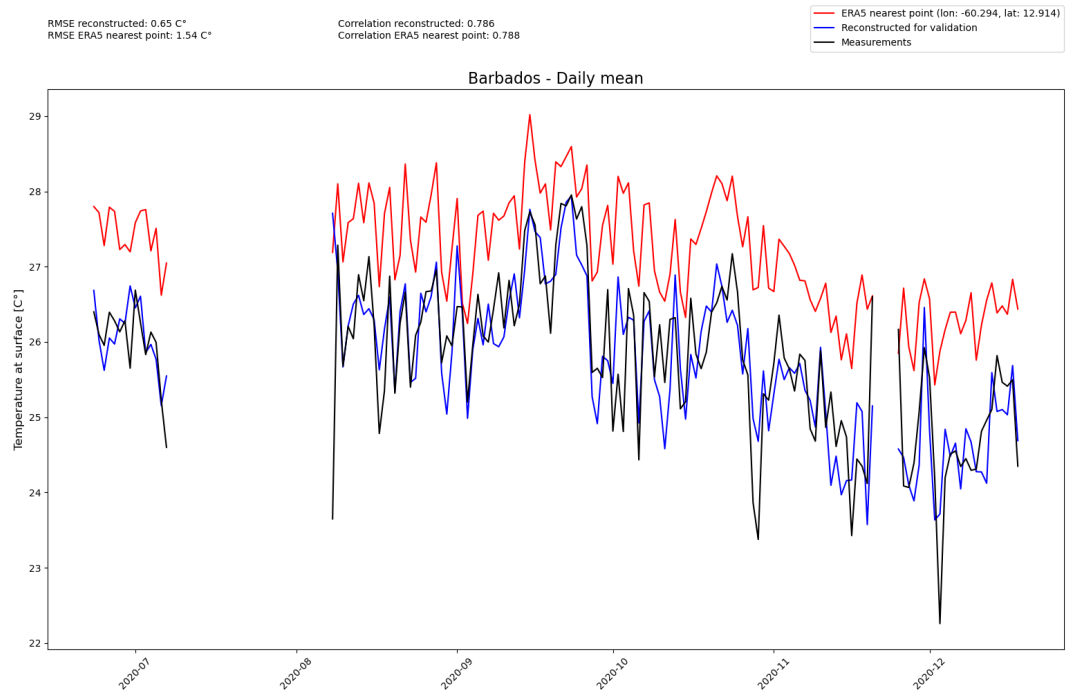



Figure 15: Reconstructed temperature for Barbados-Station (Daily mean)

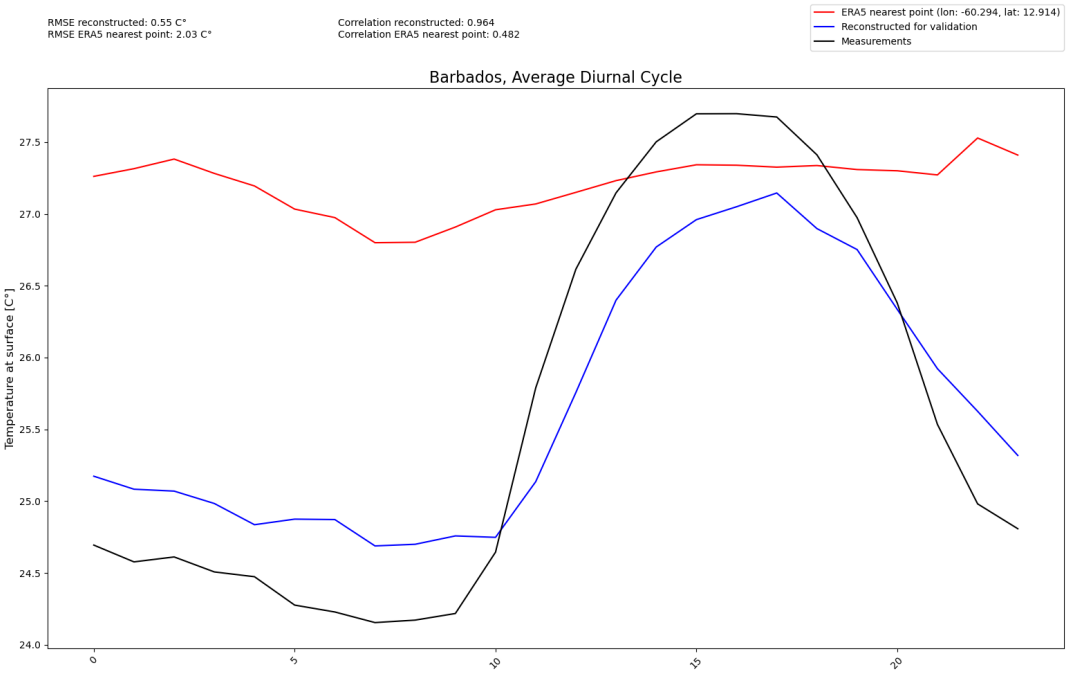



Figure 16: Measured temperature for Barbados-Station (Average Diurnal Cycle)

### 5.4 Experimenting with time context

### 5.5 How to improve results

# 6 Software Implementation

## 6.1 Introduction

The "Climate Reconstruction AI" (CRAI) Module is used, such that the actual setup of the neural net, training and evaluation afterwards is outsourced. Thus the process for a single specific dataset of one station could be written pretty straight forward with a NetCDF file of the station data ahead, if there is sufficient access to ERA5 Data. A jupyter notebook could be sufficient as a way to start. However once dealing with different stations, different ERA5 files need to be stored, and the failes that have been prepared for submission into CRAI need to have some kind of management and the "trainings-args" for CRAI need to be adapted each time. Thus it appears natural to implement a set of functions and structure the process through an object oriented approach. This allows to control different but similar pipelines with a few lines of code, either through a script, initiated by an api or in a jupyter notebook. The user then needs to take minimal care of the file management and temporary folders, as well and it is quite straightforward to pass the different objects its input and output data. When the functions are outlined in the follwing sections, it might not be explicitly stated what temp folders or output folders are created but it is a crucial feature of the implementation.

## 6.2 Stationdata Submission & 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 like the station name, the latitude and longitude and the elevation. The data files (.dat files) hold per sensor one column and per minute one row.
A class "DatToNcConverter" is implemented with the following main use cases: First to take in a directory with the .dat files and the metadata file and parse it. Second to process the data which is easiest in a pandas dataframe, dropping missing values and resampling to an hourly frequency, and converting from Celcius to Kelvin to match ERA5. Third to convert the data to a NetCDF file, which is the format that is used by the ERA5 data and the CRAI module. The NetCDF file is structured in a way that the data is stored in a 3D array with the dimensions time, latitude and longitude. The metadata

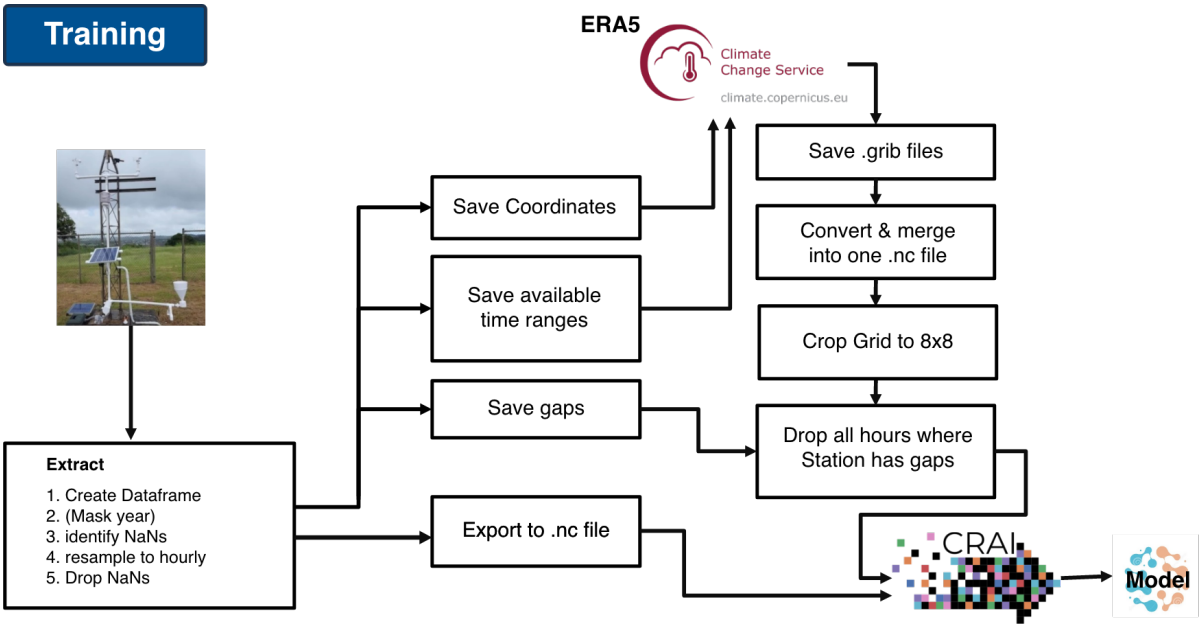



Figure 17: Pipeline to train a model

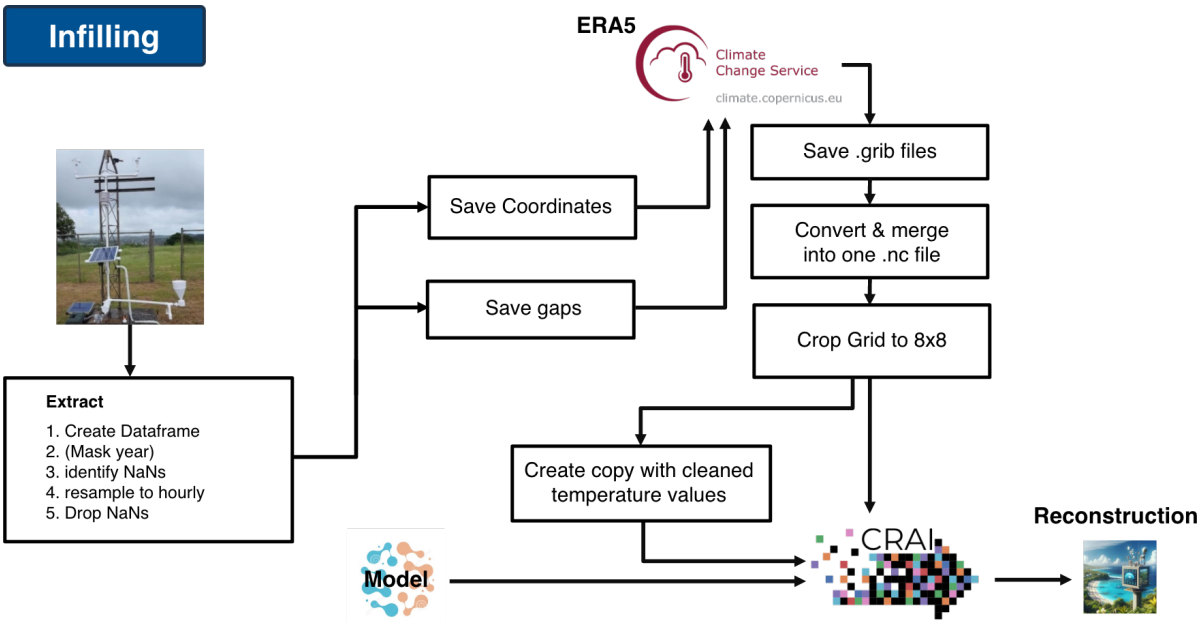



Figure 18: Pipeline to reconstruct weather data using a model

is stored as global attributes in the NetCDF file. Fourth to convert some dataframe back to a .dat file after the reconstruction of measurements to match the original format when infilling. Of these use cases the one resampling to hourly frequency is the most complex, or at least could include many design decisions. Missing values are marked with "-999.99", so in the first step, these values can be marked as NaN. However, the data quality is by default not controlled 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 anyway, but even for regions where 0°C degrees are reached often, it's unlikely to measure exactly "0.00", meaning the amount of correct data lost through marking "0.00" as NaN is still limited. Also by agreement with the NCAR everything above or under +/- 45°C is marked as NaN. Additionally to compensate for peaks in the data aggregating the minutes using a median instead of a mean can be a good idea, numpy.median() would return NaN if any value is NaN and numpy.nanmedian() would ignore NaNs. It would be best to have a custom aggregate function using numpy.nanmedian() but assuring prior to that that there are sufficient non-NaN values.
For the usecase of converting the data back to a .dat file, the dataframe is stored directly in the converter object as original dataframe after detecting NaNs and resampling to hourly values, before any transformation of units (Celcius to Kelvin) or renaming of columns takes place and before NaNs are dropped. Fundamentally inbetween the first and last measurements rows for all minutes exist, even if measurements are missing. However if the station had severe issues it's possible that between first and last record there are even rows or files missing. To assure that the original dataframe will have rows for all hours, a handy method provided by the python pandas module is used:

```
1     self.original_df = self.original_df.reindex(
2         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 holds the converter itself, to minimize lines of code in the main script. The class is mainly used to manage the access to the different files and data, before and after the conversion. And secondly to detect the gaps in the data, for infilling simply in the form of listing the hours where data is missing. And for training use cases additionally in form of listing the months where at least some data is available which is most convenient for the API applicance to get ERA5 data for the station.

```
1 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 opensource 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

```

```
12     def _download(cds, date_info, save_to_file_path):
13         cds.retrieve(
14         'reanalysis-era5-single-levels',
15         {
16             "product_type": "reanalysis",
17             "format": "grib",
18             "variable": "2m_temperature",
19             "area": [
20                 self.lat + 1, # limit north
21                 self.lon - 1 % 360, # limit west
22                 self.lat - 1, # limit south
23                 self.lon + 1 % 360, # limit east
24             ],
25             "year": date_info.get("years"),
26             "month": [f"{month:02d}" for month in date_info.get("months")],
27             "day": [f"{day:02d}" for day in date_info.get("days")],
28             "time": [f"{hour:02d}:00" for hour in date_info.get("hours")]
29 
30         }, save_to_file_path)
```

Code Snippet 3: Download Hook for ERA5 Data

As seen in the code snippet, the request needs to include besides basic informations such as variable / format / model, the regional selection and the selection in the time dimension. to download a few hours in a day the following can be used. When building requests that download a month or a full year, it becomes obvious why the Code Snippet 2 is so useful.

```
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

In the Code Snippet 3 it can be seen that the regional selection is always +/- 1 degree around the Station location. This ensures because of the grid interval of 0.25° 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 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

```
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:

```
    import xarray as xr
    import subprocess

    def _rename_variable(self, var_name, tas_name, input, output):
        rename_variable_command = \
            f"cdo chname,{var_name},{tas_name} {input_path} {output_path}"
        subprocess.run(rename_variable_command, shell=True)

    ds = xr.open_dataset(input_path)
    if 'var167' in ds.variables:
        _rename_variable('var167', 'tas', input_path, output_path)
    elif '2t' in ds.variables:
        _rename_variable('2t', 'tas', input_path, output_path)
```

Code Snippet 5: Renaming Variable in NetCDF File

Then the files are merged using

```
    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 / 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 usinfg 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 coppies 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, in a validation procedure the same preparation of ERA5 data is done, 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 NaNs.
When evaluating the model not for validation but to infill missing values, the ERA5 data needs to be prepared differently of course. Instead of requesting monthly data from the cdsapi, it is requesing only the missing hours day by day, such that the timedimesion is directly as desired, and preprocessing only needs to handle conversion and geographical cropping.

## 6.5 CRAI - Climate Reconstruction AI

Because CRAI is using the encode, decode architecture with connected layers 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 has been trained on input files where the groudtruth was also laid on the 8x8 grid, the 64 temperature values in the output data of the model tend to be extremely similar. They need to be reshaped however to the original dimensions of the station data, which is a 1D array with 1 temperature value per timestep. This is done by taking the mean of the 64 values.

# 7 Process Orchestration + API & Webinterface

## 7.1 Executer Classes, (Training, Infilling, Validation)

## 7.2 API Endpoints

## 7.3 Webinterface

# 8 Discussion

## 8.1 How much data is needed?

## 8.2 Use for weather forecasting

# References

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

[2] Ariel Ortiz-Bobea. Climate, agriculture and food, 2021.

[3] R. Marchant, C. Mumbi, S. Behera, and T. Yamagata. The indian ocean dipole—the unsung driver of climatic variability in east africa. *African Journal of Ecology*, 45:4–16, 2007. doi: 10.1111/j.1365-2028.2006.00707.x.

[4] 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.

[5] 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.

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

[7] Mark W. Williams, Mark Losleben, Nel Caine, and David 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.