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 18, 2024
Place, Date, Signature // Ort, Datum, Unterschrift

## Abstract

## Zusammenfassung

# 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. [3] 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. Not only would a denser more reliable network benefit weather forecasting, but it would also be beneficial for 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. [1, 2] 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. [2], either way, low-cost weather stations have reliability issues.

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 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 blurry and describes the weather in grid cells, 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. The reanalysis of choice 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, 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.

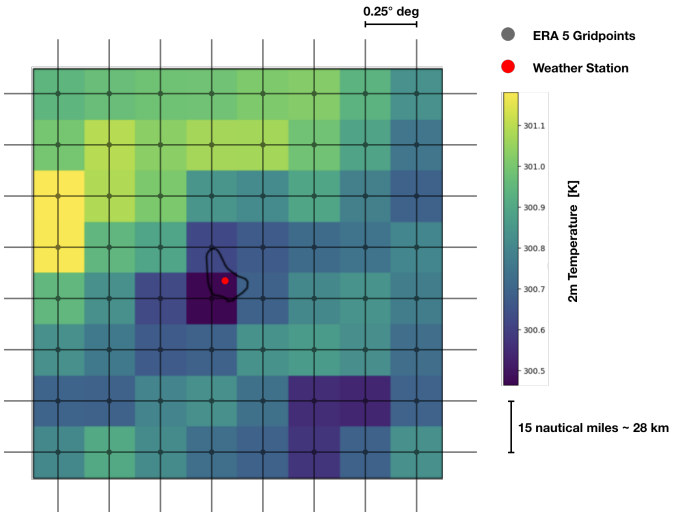


Figure 1: 8x8 grid-points of ERA5 with 2m temperature for 2020-06-23 19:00 UTC in the area of a weather station on Barbados

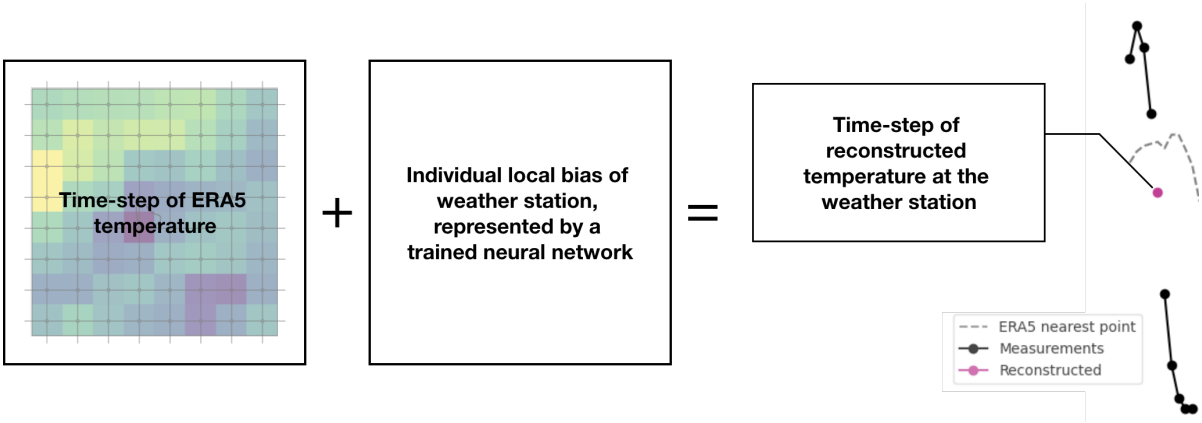


Figure 2: Conceptual Framework of the proposed method

# 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, 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

**Convolutional Layer**

**Pooling Layer**

**Fully Connected Layer**

**Activation Function**

**Loss Function**

**Optimization Algorithm**

## 4.2 Reanalysis - ERA5

## 4.3 Weather Station Data Quality

# 5 Results

## 5.1 Results of basic setup

## 5.2 Experimenting with time context

## 5.3 How to improve results

# 6 Software Implementation

## 6.1 Introduction

The "Climate Reconstruction AI" 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

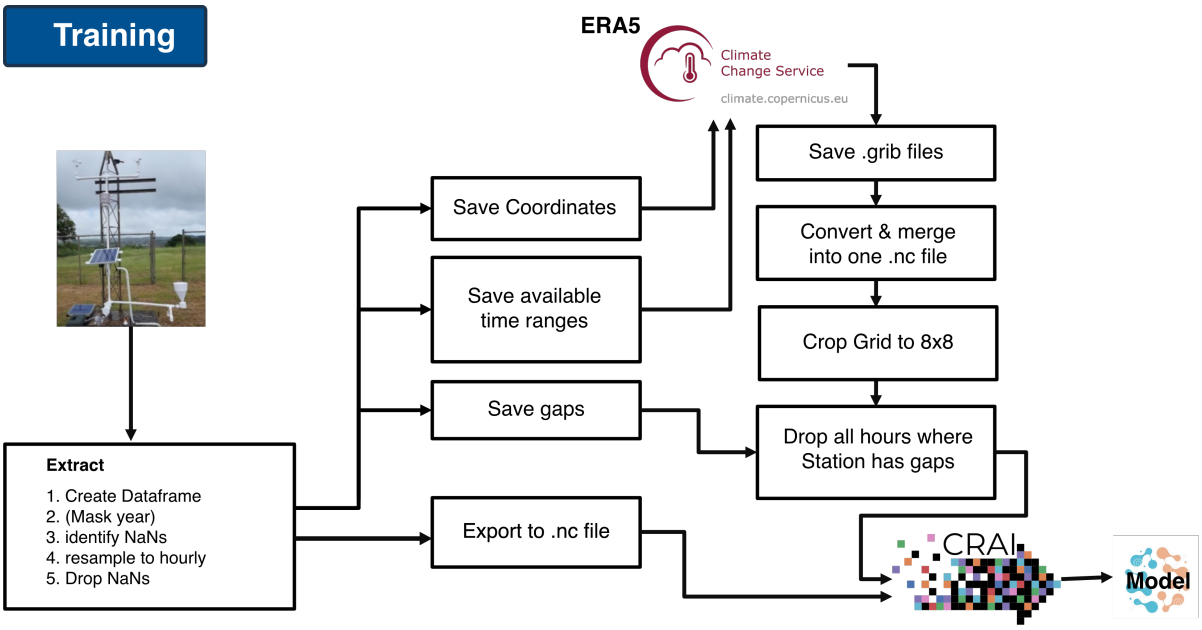


Figure 3: Pipeline to train a model

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 temporarly folders, as well and it is quite straight forward 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 is stored as global attributes in the NetCDF file. Fourth to convert some dataframe

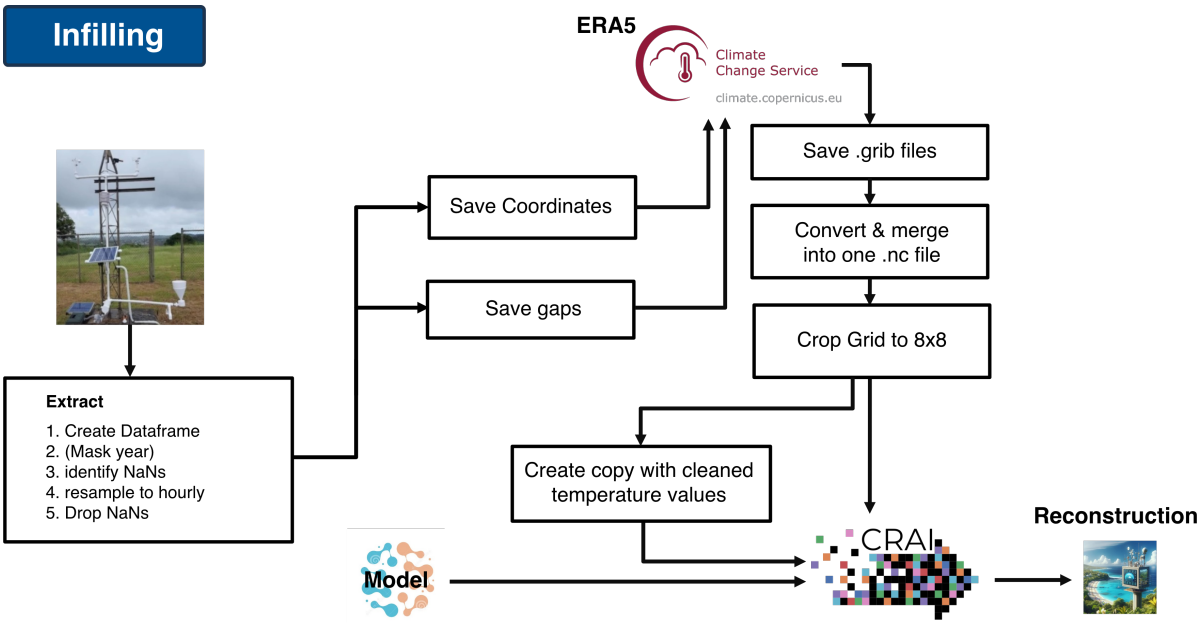


Figure 4: Pipeline to reconstruct weather data using a model

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:

```
    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 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.

```
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.

```python
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")]

        }, 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.

```
    def download_hours_in_same_day(self, year, month, day, hours, ↩
        target_folder):
        self.download({
            "years": [year],
            "months": [month],
            "days": [day],
            "hours": hours
        }, 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:

```
    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)

    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)
```

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.

### 6.6 Executer Classes, Training, Infilling, Validation

### 6.7 API & Webinterface

## 7 Discussion

### 7.1 How much data is needed?

### 7.2 Use for weather forecasting

## References

[1] 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.

[2] R. Muita, P. Kucera, S. Aura, D. Muchemi, D. Gikungu, S. Mwangi, M. Steinson, P. Oloo, N. Maingi, E. Muigai, and M. Kamau. Towards increasing data availability for meteorological services: Inter-comparison 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.

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