



Global Earth Monitor



Deliverable 2.4

Meteo/climate service





PREPARATION SLIP			
	Name	Organisation	Date
Prepared by	Nicoletta Addimando, Patrick Zippenfenig	meteoblue	26/08/2021
Prepared by	Matej Aleksandrov	Sinergise	28/08/2021
Reviewed by	Matej Batič	Sinergise	30/08/2021
Approved for submission by	Matej Batič	Sinergise	31/08/2021
Reviewed by the EC		EC	28.01.2022
Comments addressed by	Nicoletta Addimando	meteoblue	15.02.2022
Comments addressed by	Matej Batič	Sinergise	22.02.2022
Approved for resubmission	Mariza Pertovt	Sinergise	24.02.2022
Last iteration content prepared by	Christoph Ramshorn Frederic Schwarz Sebastian Schlögl Nico Bader	meteoblue	24.08.2023
Reviewed and approved for resubmission	Matej Batič	Sinergise	29.08.2022

EXECUTIVE SUMMARY
<p>This is the 3rd updated report on the meteoblue meteo/climate services created or optimised as part of the GEM project.</p> <p>Model data and measurements are made available to GEM project and public through meteoblue dataset API and measurements API, described in this report. An adapter has moreover been developed to access meteoblue data directly through the Sinergise eo-learn library, bringing Earth Observation and weather data together. A temperature downscaling to 10 m resolution for built-up areas has been developed.</p> <p>This deliverable provides moreover a description of:</p> <ul style="list-style-type: none">• new measurements sets added during the course of the project and the implemented gridding methodology;• description of the eo-learn adapter;• new functionalities added to the dataset API to serve machine learning applications;• gridded operations designed to seamlessly access multi-resolution models;• using the meteoblue advanced API to obtain climate variables;• API access to high-resolution (10m) temperature fields in built-up areas. <p>Keywords: Weather data, API, eo-learn, temperature downscaling</p>

Contractual Delivery Date	31.08.2021
Actual Delivery Date	31.08.2021
Delivery Date for version v2	24.02.2022
Delivery Date for version v3	31.08.2021
Type of delivery	Public
Dissemination level:	Demonstrator

Table of Contents

1 Introduction.....	1
2 Weather data.....	2
2.1 Model data.....	2
2.2 Measurement data	2
3 Service enhancements	6
3.1 Dataset SDK improvements	6
3.2 Gridded operations with multi-resolution models.....	8
3.3 Derived weather variables.....	9
3.4 eo-learn adapter	17
3.5 Optimisation of meteoblue nowcasting and forecasting services	23
3.6 High resolution temperature fields API	28
Conclusions.....	35

List of Figures

Figure 1 : Example of measurement API response.	3
Figure 2: meteoblue measurements interface	4
Figure 3: Output example of a measurement API query.	5
Figure 4: Selected location and surrounding grid cells.	6
Figure 5: Recommended approaches for feature composition	7
Figure 6: Model domains available at meteoblue	8
Figure 7: Minimum and maximum elevations (in meters) of ERA5T grid cells in Ticino.	17
Figure 8: Comparison of nearest-neighbour selection and bilinear interpolation (Cmglee, CC BY-SA 4.0).	18
Figure 9: Resampling with temperature downscaling.	20
Figure 10: Example of MeteoblueVectorTask.....	21
Figure 11: Resulting weather (tabular) data from the MeteoblueVectorTask.....	21
Figure 12: Example of MeteoblueRasterTask from eo-learn	22
Figure 13: Single EO observation with corresponding weather data.....	22
Figure 14: meteoblue Learning Multimodel process chain	23
Figure 15: ArchiMate diagram of Nowcasting Pipeline within meteoblue API	25
Figure 16: System architecture diagram of the integration of the meteoblue City Climate Model (mCCM) in the process chain of the meteoblue core system.	28
Figure 17: Components of the mCCM and their verifications.	28
Figure 18: Example of a heat map output for Berlin for 2023-07-13T07:00.	31

List of Abbreviations

GEM	Global Earth Monitor
NWP	Numerical Weather Prediction
EO	Earth Observation
ML	Machine learning
API	Application Programming Interface
STAC	Spatio Temporal Asset Catalog
ECMWF	European Centre for Medium-Range Weather Forecasts
NOAA	National Oceanic and Atmospheric Administration
KNM	Koninklij Nederlands Meteorologisch Insituut
JMA	Japan Meteorological Agency
DWD	Deutscher Wetterdienst
CMCC	Centro euro-Mediterraneo sui Cambiamenti Climatici
BOM	Bureau of Meteorology
SatCen	European Union Satellite Centre
NetCDF	Network Common Data Form
CSV	Comma Separated Values
XLSX	Excel Open Xml Format
JSON	JavaScript Object Notation
SDK	Software Development Kit
EU	European Union
EUXDAT	European e-Infrastructure for Extreme Data Analytics in Sustainable Development
HTTP	HyperText Transfer Protocol
Madis	Meteorological Assimilation and Data Ingest System
METAR	METEorological Aerodrome Report
Synop	surface synoptic observations
UTM	Universal Transverse Mercator

1 Introduction

meteoblue meteo/climate service supports GEM project and the public by providing easy access to meteoblue model data and measurements database, through the development of apposite Application Programming Interfaces (APIs).

Data can be retrieved both from user-friendly interfaces and directly from programming environments thanks to the meteoblue Python Software Development Kit (SDK). Furthermore, a newly developed EOTask allows weather data to be retrieved from GEM processing framework eo-learn, collection of open-source Python packages that bridges between the Earth Observation (EO)/Remote Sensing (RS) domain and Python ecosystem.

The service and usage examples are described in the following chapters.

2 Weather data

2.1 Model data

meteoblue operates a large number of numerical weather prediction models and integrates open data from various sources.

Weather models divide the world or a region into small “grid-cells”. Each cell is ranging from 4 km to 40 km wide with height between 100 m and 2 km. The available models contain 60 atmospheric layers and reach deep into the stratosphere at 10-25 hPa (60 km altitude).

Each model is updated with a frequency varying from monthly to 2 times per day. The output is stored in the meteoblue weather archive (seasonal forecasts excluded) and is accessible through a web interface and a dataset API.

2.1.1 Data description

A summary of the available weather models and their characteristics can be found in the Data Management Plan (D2.1, Section 3.2).

2.2 Measurement data

2.2.1 Data description

Available measurement data include air temperature, dewpoint, windchill, soil temperature, relative humidity, windspeed, wind gust speed, wind direction, precipitation, precipitation rate, global radiation, UV index, snow depth and cloud cover. Which data are available from an individual weather station depends on that station.

2.2.2 Measurement API

During GEM project, a prototype API was finalized to allow easy access to the meteoblue measurements database.

The response retrievable from the API is in a long columnar format, thus enabling the access of long time series for one station as well as single timesteps for multiple stations at once. The user can easily limit the API response to specific fields of interest, specific stations, or a defined geographical area. Moreover, the output format can be adapted to the needs of the user.


```

{
  "current_page": 1,
  "rows": 10,
  "rows_per_page": 10,
  "columns": [
    {
      "column": "id",
      "values": [
        "05280",
        "05158",
        "05149",
        "05097",
        "05086",
        "04990",
        "04909",
        "03829",
        "04559",
        "04609"
      ]
    },
    {
      "column": "lat",
      "values": [
        53.9224,
        52.1601,
        49.5741,
        54.0661,
        51.5776,
        48.3277,
        48.8449,
        51.737,
        49.1644,
        50.1926
      ]
    },
    {
      "column": "lon",
      "values": [
        10.2267,
        11.1759,
        10.1915,
        12.7675,
        9.4335,
        12.1712,
        8.545,
        10.2456,
        12.6175,
        11.8932
      ]
    },
    {
      "column": "timestamp",
      "values": [
        "2021-05-18T00:00:00Z",
        "2021-05-18T00:00:00Z",
        "2021-05-18T00:00:00Z",
        "2021-05-18T00:00:00Z",
        "2021-05-18T00:00:00Z",
        "2021-05-18T00:00:00Z",
        "2021-05-18T00:00:00Z",
        "2021-05-18T00:00:00Z",
        "2021-05-18T00:00:00Z",
        "2021-05-18T00:00:00Z"
      ]
    },
    {
      "column": "precip_1mAbvGnd_prevDay_total_mm",
      "values": [
        0.1,
        0.3,
        0.3,
        0.1,
        0.3,
        3.8,
        14.3,
        null,
        3.4,
        1.8
      ]
    }
  ]
}

```

Figure 1 : Example of measurement API response.

2.2.3 meteoblue Measurement Interface

To enable an easy access to the meteoblue measurements API meteoblue provides a [user interface](#) that allows near real-time access to the available measurements and enables users to easily find and select stations which fulfil defined criteria. The users can:

- have an overview over available stations
- select stations within a specific geographic area represented by polygons
- select stations with a minimum averaged hourly coverage regarding a specific weather variable during a time interval
- configure data downloads

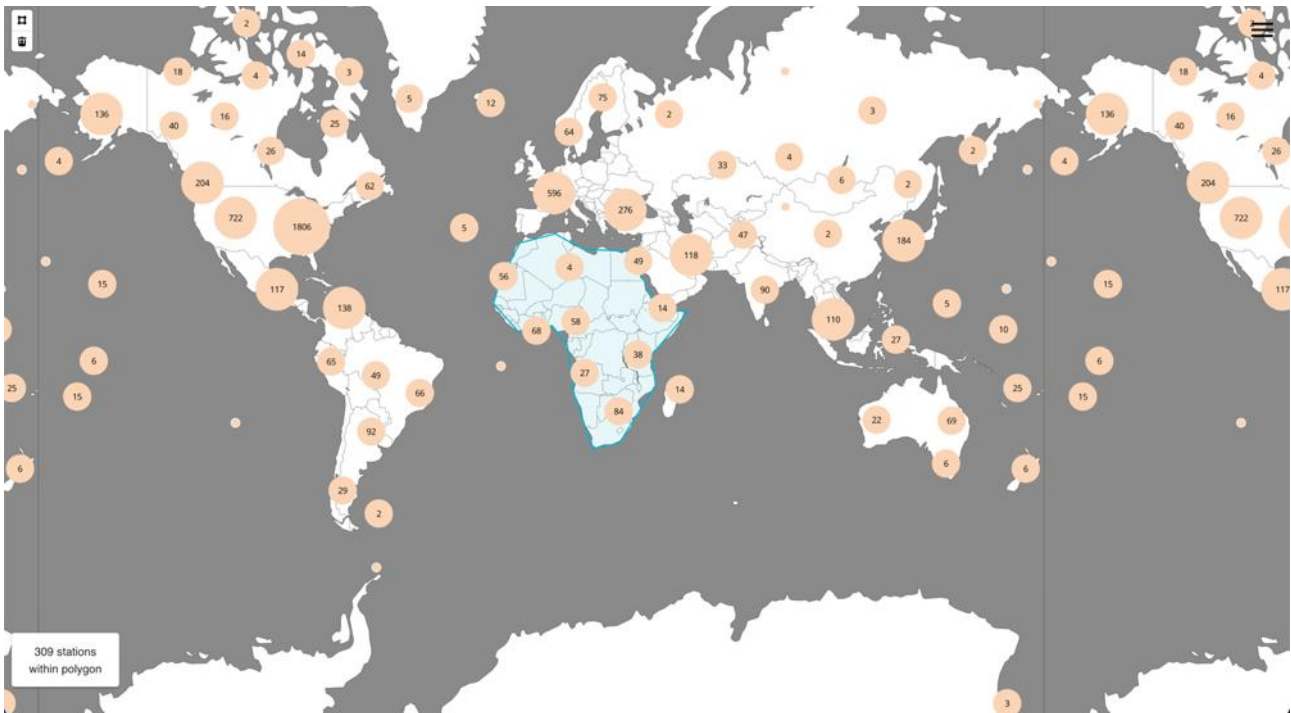


Figure 2: meteoblue measurements interface

2.2.4 meteoblue Measurement SDK

meteoblue Python SDK was enhanced to provide access to measurement data retrieval directly from Python programming environment.

Tutorials with usage examples in the form of Jupyter notebooks and Python files are available (for partners of GEM project) in [GEM project repository](#) and publicly published in meteoblue [GitHub repository](#).

```

## query measurement data for time interval
path = "/rawdata/dwdClimateHourly/dwdClimateMeasurementHourlyAirTemperature/get"
name_variable_measured = "temperature_2mAbvGnd_atTimestamp_none_degCels"
query = {
  "timeStart": "2020-01-01T00:00:00",
  "timeEnd": "2020-12-31T23:00:00",
  "limit": 10000,
  "fields": [
    "id",
    "timestamp",
    "lat",
    "lon",
    "asl",
    name_variable_measured
  ],
  "stations": ["00691"],
  "sort": "asc"
}

client = meteoblue_dataset_sdk.Client(apikey=os.environ["APIKEY"]) # apikey has to be saved
measurements_queried = client.measurement_sync(path, query)
measurements_df = measurements_to_dataframe(measurements_queried)

# set coordinates and station_name for request of modeled data
# this only works if only one station is selected in the above request
lat = measurements_df["lat"][0]
lon = measurements_df["lon"][0]
asl = measurements_df["asl"][0]
station_name = measurements_df["id"][0]

```

	id	timestamp	lat	lon	asl	temperature_2mAbvGnd_atTimestamp
0	00691	1577836800	53.044998	8.7979	4.28	-1.8
1	00691	1577840400	53.044998	8.7979	4.28	-0.9
2	00691	1577844000	53.044998	8.7979	4.28	-2.2
3	00691	1577847600	53.044998	8.7979	4.28	-3.3
4	00691	1577851200	53.044998	8.7979	4.28	-2.7
...
8779	00691	1609441200	53.044998	8.7979	4.28	2.2
8780	00691	1609444800	53.044998	8.7979	4.28	1.0
8781	00691	1609448400	53.044998	8.7979	4.28	1.2
8782	00691	1609452000	53.044998	8.7979	4.28	1.2
8783	00691	1609455600	53.044998	8.7979	4.28	1.3

Figure 3: Output example of a measurement API query.

3 Service enhancements

3.1 Dataset SDK improvements

meteoblue Dataset SDK has been improved to support the development of the eo-learn adapter and to better serve machine learning applications.

3.1.1 General improvements

In co-development to provide the eo-learn adapter, the Dataset SDK was improved to work with common data science libraries (e.g., GeoPandas for geospatial data).

In addition, a data cache was developed to improve development for machine learning models with usually require multiple execution cycles to optimize model parametrization. The cache feature was made available not only to eo-learn, but also to other users of the meteoblue Dataset SDK.

3.1.2 Surrounding grid cells

Machine learning applications may benefit from adding the information not only from the point value of a weather variable, but also from values in the surrounding grid cells (Figure 4). This is rather important also to prevent border effects when requesting data in a tile manner (e.g., non-overlapping EO patches).

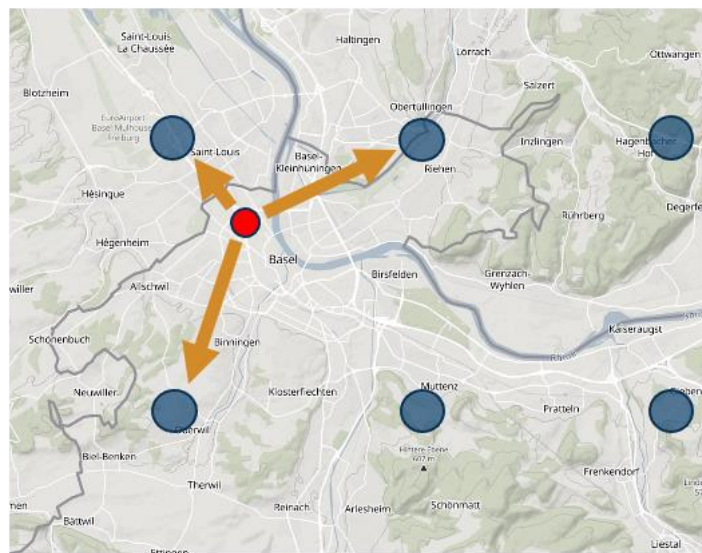


Figure 4: Selected location and surrounding grid cells.

Therefore, the functionality to obtain the values in the surrounding grid cells of a selected location was implemented: it can be done automatically from both the meteoblue web interface and meteoblue dataset SDK (Figure).

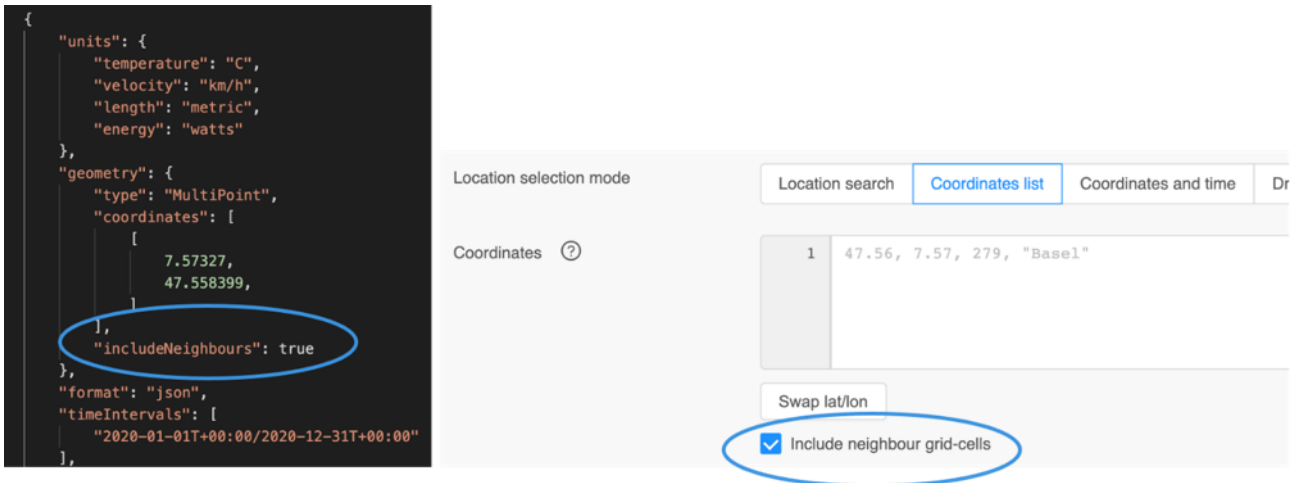


Figure 5: Surrounding grid cell functionality in dataset query and option in meteoblue web interface.

3.1.3 Feature composition

As recommended in deliverable D4.1 (Modelling requirements), the best ways to deal with two datasets of a coarser (A) and a finer (B) temporal resolution are the systematic approaches, where each time instance of A is expanded with the same number of instances of B (Figure 5).

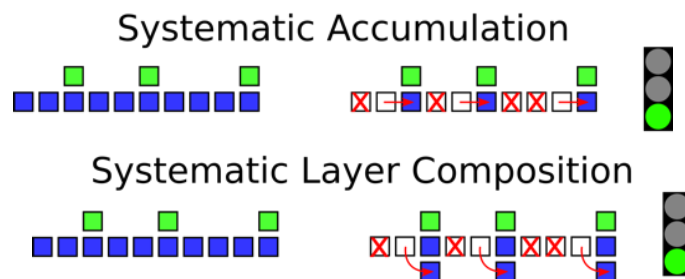


Figure 5: Recommended approaches for feature composition (see deliverable 4.1 for thorough explanation).

In the GEM project, the use cases typically deal with:

- Earth Observation (EO) datasets of coarser temporal resolution (5-15 days)
- Weather datasets of finer temporal resolution (hourly or daily)

A library, currently available in GEM project repository, has therefore been developed to systematically associate weather data with each EO time instance by following the two recommended approaches of Figure 5.

The user can perform systematic accumulation by averaging or summing the selected weather variables, starting from several time instances before the EO that can be tailored for each weather variable but remains consistent along all the considered EO times. Examples include summing, for each EO instance, the precipitation of one month before, or averaging the maximum temperatures over the previous two weeks.

Time series of weather data can also be added as an additional layer (systematic layer composition) instead of being accumulated: this way the whole temporal evolution is stored, and more information preserved, although the dimensionality of the dataset (and complexity of the model) increases.

This approach introduces expert knowledge by controlling:

- a. which EO data and weather variables to select for a specific use case
- b. what time window to consider such that each weather variable is most representative for a specific application.

3.2 Gridded operations with multi-resolution models

meteoblue offers weather models in various regions at different spatial resolutions (Section 2.1).

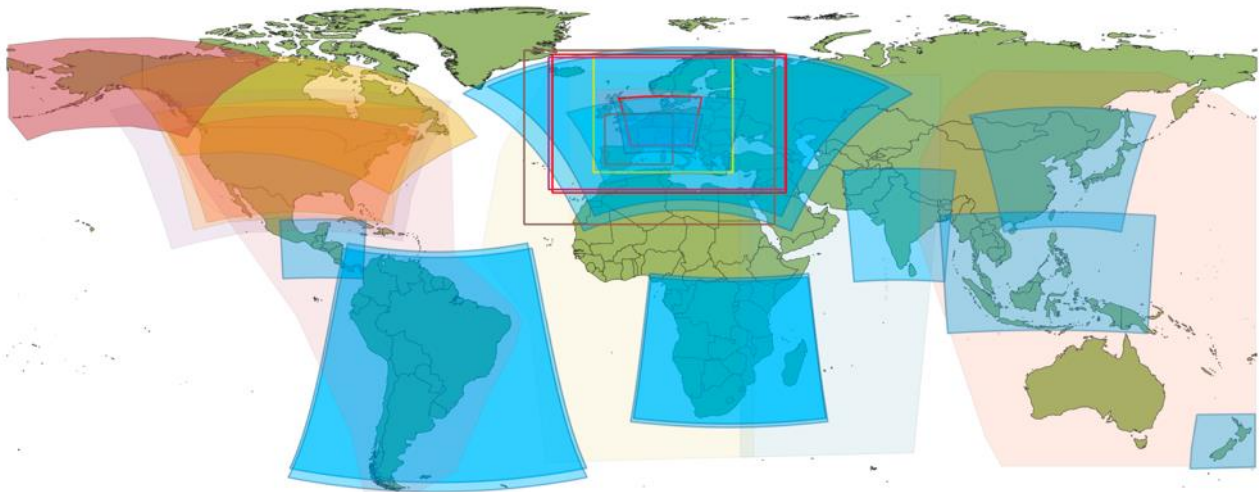


Figure 6: Model domains available at meteoblue (global models not shown). In blue meteoblue NEMS and NMM numerical weather model families.

To exploit the availability of higher resolution models for certain areas (e.g., NEMS4-NEMS8-NEMS10-NEMS12-NEMS30 or ICON7-ICON), a chain of on-the-fly operations is performed to allow a seamless and consistent transition between models.

The higher resolution models are integrated in the lower resolution ones by chaining bilinear interpolation (Section 3.4.1), domains overlay, and borders smoothing to avoid boundary effects and data inconsistencies.

This processing chain is implemented to, for example, integrate weather data in GEM's Adjustable Data Cubes.

3.3 Derived weather variables

In discussions with project partner the need for derived weather and climate variables became apparent. Derived here means that several "raw" variables need to be combined to obtain a desired variable. meteoblue supports this through a calculation engine in its Dataset API and, by extension, through its Python SDK and eo-learn adapter (Section 3.3). The calculations of derived variables benefit from the optimisations described in Section 3.5, making them more practical for larger datasets.

The following sections describe the derived variables and their calculation. The first example provides all technical details, which are the same for all derived variables save a few parameters. Full examples for all derived variables are given in Jupyter Notebooks¹.

3.3.1 Climate Indices

“A climate index is defined as a calculated value that can be used to describe the state and the changes in the climate system” - [Integrated Climate Data Center \(ICDC\)](#).

3.3.1.1 Tropical Nights

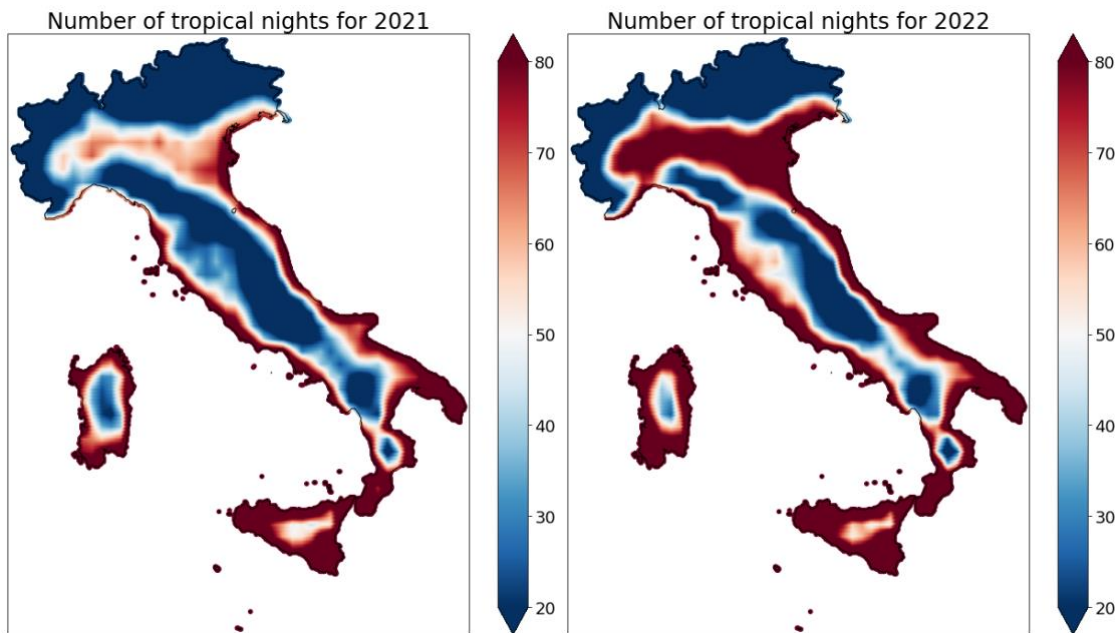
A tropical night is a term used to describe days when the temperature does not fall under 20 °C during the night-time. Due to climate change, many countries are experiencing a significant increase in tropical nights compared to last century.

Query for yearly number of tropical nights for a given location:

1. Select a location and a time interval of 1 or more years.
2. Select the dataset ERA5T, Temperature, 2 m above gnd, hourly.
3. Aggregate by day to daily **minimum**.
4. Value is greater than or equal to **20** then pick **0 or 1**.
5. Aggregate by year to yearly **summation**.

Note: if producing a map or if the ERA5T grid-cell is in the immediate vicinity of the sea remember to mark the option “exclude sea points”.

¹ https://github.com/sentinel-hub/eo-learn-examples/tree/main/climatological_days



The graphs above were obtained through Python, simply by selecting from the Dataset API a geographic polygon (in this case Italy), instead of a single location. The following option included in the available transformations was also used to obtain a better representation:

"Resample to a regular grid using linear interpolation", with a grid resolution of 0.025°.

Using the `meteoblue_dataset_sdk` Python package it is possible to retrieve data for creating the above maps in a few simple steps, defining a unique query. The tropical nights' example is shown in the following. All other examples are formed analogously.

Query for tropical nights:

```
query_map = {
  "units": {
    "temperature": "C",
    "velocity": "km/h",
    "length": "metric",
    "energy": "watts"
  },
  "geometry": {
    "type": "GeonamePolygon",
    "geonameid": 3175395 #this code corresponds to Italy
  },
  "format": "JSON",
  "timeIntervals": [
    str(year)+"-01-01T+00:00/"+str(year)+"-12-31T+00:00"
  ],
  "timeIntervalsAlignment": "none",
  "queries": [
    {
      "domain": "ERA5T",
      "gapFillDomain": None,
      "timeResolution": "hourly",
```



```

"codes": [
  {
    "code": 193,
    "level": "2 m above gnd"
  }
],
"transformations": [
  {
    "type": "aggregateDaily",
    "aggregation": "min"
  },
  {
    "type": "valuesAbove",
    "valueMin": 20,
    "returnClassification": "zeroOrOne"
  },
  {
    "type": "aggregateYearly",
    "aggregation": "sum"
  },
  {
    "type": "spatialTransform",
    "gridResolution": resolution,
    "interpolationMethod": "linear",
    "spatialAggregation": "mean",
    "disjointArea": "discard",
    "elevationDownscale": "disabled"
  }
]
}
]
}

```

API call:

```
client = meteoblue_dataset_sdk.Client(apikey)
```

```
result = client.query_sync(query_map)
```

```
lats = list(result.geometries[0].lats)
```

```
lons = list(result.geometries[0].lons)
```

```
#Use a function to convert the datasetAPI protobuf file into a pandas dataframe
```

```
data = list(meteoblue_result_to_dataframe(result.geometries[0])["193_2 m above gnd_sum"])
```

3.3.1.2 Frost Days

Frost days are defined as days in which temperature drops below 0°C.

Query for yearly number of frost days for a given location:

1. Select a location and a time interval of 1 or more years.
2. Select the dataset ERA5T, Temperature, 2 m above gnd, hourly.
3. Aggregate by day to daily **maximum**.

4. Value is less than or equal to **0** then pick **0 or 1**.
5. Aggregate by year to yearly **summation**.

Note: if producing a map or if the ERA5T grid-cell is in the immediate vicinity of the sea remember to mark the option “exclude sea points”.

3.3.1.3 Ice Days

Ice days are defined as days in which temperature never goes above 0°C.

Query for yearly number of ice days for a given location:

1. Select a location and a time interval of 1 or more years.
2. Select the dataset ERA5T, Temperature, 2 m above gnd, hourly.
3. Aggregate by day to daily **maximum**.
4. Value is less than or equal to **0** then pick **0 or 1**.
5. Aggregate by year to yearly **summation**.

Note: if producing a map or if the ERA5T grid-cell is in the immediate vicinity of the sea remember to mark the option “exclude sea points”.

3.3.1.4 Hot Days

Hot days are defined as days in which temperature rise above 30°C.

Query for yearly number of hot days for a given location:

1. Select a location and a time interval of 1 or more years.
2. Select the dataset ERA5T, Temperature, 2 m above gnd, hourly.
3. Aggregate by day to daily **maximum**.
4. Value is greater than or equal to **30** then pick **0 or 1**.
5. Aggregate by year to yearly **summation**.

Note: if producing a map or if the ERA5T grid-cell is in the immediate vicinity of the sea remember to mark the option “exclude sea points”.

3.3.1.5 Heating Degree Days (HDD) & Cooling Degree Days (CDD)

Heating degree days are a measure of how much (in degrees), and for how long (in days), the outside air temperature was below a certain level. They are commonly used in calculations relating to the energy consumption required to heat buildings. With regard to heating degree days, the *base temperature* of a building is the outside air temperature below which that building needs heating.

Cooling degree days are a measure of how much (in degrees), and for how long (in days), the outside air temperature was *above* a certain level. They are commonly used in calculations relating to the energy consumption required to *cool* buildings.

Further explanations can be found at <https://www.degreedays.net/introduction>.

Hot days are defined as days in which temperature rise above 30°C.

Query for yearly number of heating degree days for a given location:

1. Select a location and a time interval of 1 or more years.
2. Select the dataset ERA5T, Temperature, 2 m above gnd, hourly.
3. Value is less than or equal to **15** then pick **0 or delta**. (Here 15 is the base temperature, which can be arbitrarily selected by the user).
4. Aggregate by year to yearly **summation**.
5. Download the file as CSV and divide the values by 24.

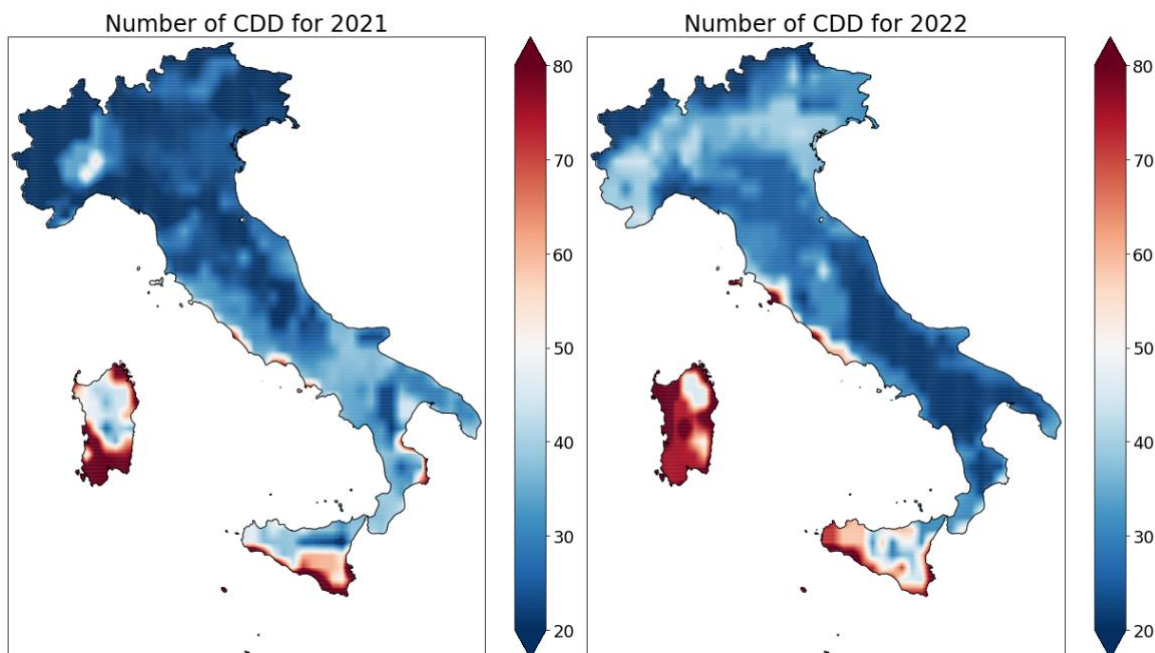
Note: if producing a map or if the ERA5T grid-cell is in the immediate vicinity of the sea remember to mark the option “exclude sea points”.

3.3.1.6 Consecutive Dry Days

Consecutive dry days are defined as the number of consecutive days with less than 1mm of rain up to a maximum of 365 (1 year). This index is really important for evaluating drought conditions.

Query for maximum number consecutive dry days in a year for a given location:

1. Select a location and a time interval of 1 or more years.
2. Select the dataset ERA5T, Precipitation, Surface, daily sum.
3. Value is less than or equal to **1** then pick **0 or consecutive count**.
4. Aggregate by year to yearly **maximum**.



The graphs above were obtained through Python, simply by selecting from the Dataset API a geographic polygon (in this case Italy), instead of a single location. The following option included in the available transformations was also used to obtain a better representation:

"Resample to a regular grid using linear interpolation", with a grid resolution of 0.025°.

3.3.1.7 Heavy Precipitation Days

Counting the number of days within a given period in which daily cumulative precipitation has exceeded a given threshold (such as 20, 30, 40 mm).

Query for number of heavy precipitation days in a year for a given location:

1. Select a location and a time interval of 1 year.
2. Select the dataset ERA5T, Precipitation, Surface, daily sum.
3. Value is greater than or equal to **20/30/40** then pick **0 or 1**.
4. Aggregate by year to yearly **summation**.

3.3.1.8 Violent Precipitation Events

Counting the number of events in one year during which the cumulative precipitation in 3 days exceeded 150 mm. This index is very important because it provides an estimate of the meteorological contribution to the hydrological risk of a given area.

Query for number of violent precipitation events in a year for a given location:

1. Select a location and a time interval of 1 year.
2. Select the dataset ERA5T, Precipitation, Surface, daily sum.
3. Aggregate over a sliding time window using a running **Summation** of the last **3** timesteps.
4. Value is greater than or equal to **150** then pick **0 or 1**.
5. Aggregate by year to yearly **Summation**.

3.3.2 Others

3.3.2.1 Atmospheric Stability

Compute the heavy storm likelihood, i.e., the number of days in which the CAPE (Convective Available Potential Energy) value is above a certain threshold (e.g., 5000 J/kg).

1. Select a location and a time interval of 1 year.
2. Select the dataset ERA5T, CAPE, 180-0 mb, daily max.
3. Value is greater than or equal to **5000** then pick **0 or 1**.
4. Aggregate by year to yearly **summation**.

3.3.2.2 Agricultural Photovoltaic (Agro PV)

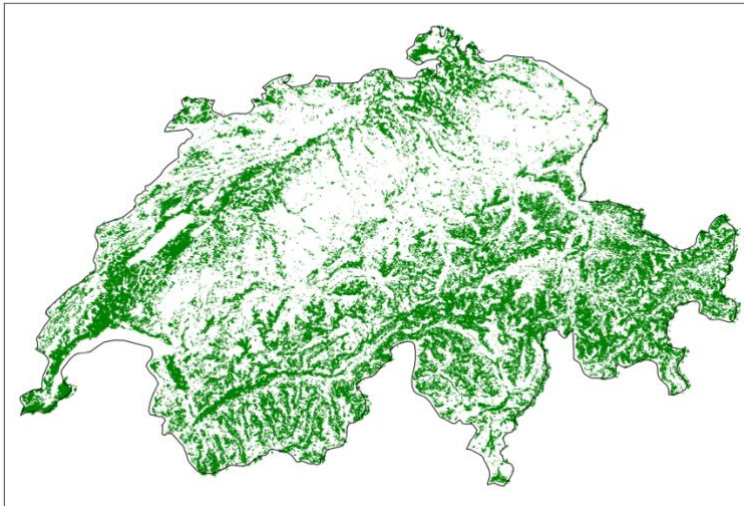
Agro PV is the simultaneous use of areas of land for both [solar panels](#) and [agriculture](#). Because solar panels and crops must share the sunlight, the design of Agro PV facilities may require trading off such objectives as optimizing [crop yield](#), crop quality, and energy production. In some cases, crop yield increases due to the shade of the solar panels mitigating some of the stress on plants caused by high temperatures and [UV](#) damage.

Thanks to the wealth of datasets provided by our API, we sought to construct a parameter expressing the suitability of a land to host Agro PV plants. The project is still under development, but the following steps give an idea of the potential of this approach.

From the GLOBCOVER dataset [0.3 km], we selected only the land cover types which may be suitable for installing PV panels. The list is (in red the one labeled as “suitable”):

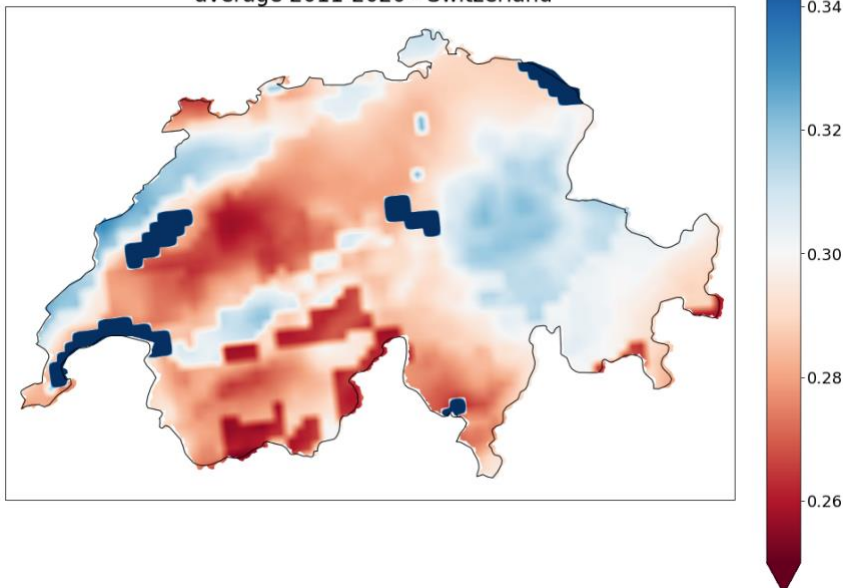
- 11 Post-flooding or irrigated croplands (or aquatic)
- 14 Rainfed croplands
- 20 Mosaic cropland (50-70%) / vegetation (grassland/shrubland/forest) (20-50%)
- 30 Mosaic vegetation (grassland/shrubland/forest) (50-70%) / cropland (20-50%)

Suitable Land Covers - Switzerland



From the NEMS4 model [4 km], we selected soil moisture 0-10 cm, averaged over a period of 10 years (2011 – 2020).

Soil Moisture 0-10 cm - NEMS4
average 2011-2020 - Switzerland

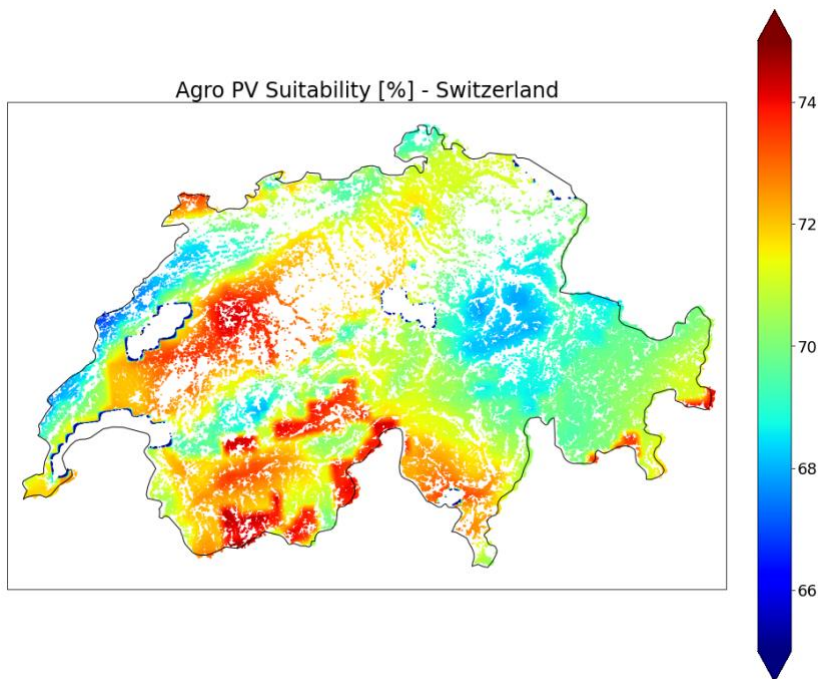


We tried to come up with a parameter for Agri PV suitability combining the previous two data frames and adding some conditions:

- If soil moisture is > 0.4 then it's not suitable: too wet, need of intense solar radiation.

- If soil moisture is < 0.2 then it's not suitable: too dry.

In the range between 0.2 and 0.4 the more suitable terrains are the drier ones (more potential improvement from agro PV, therefore define the suitability as proportional to $(1 - \text{soil moisture})$).



3.3.2.3 Difference of elevation in ERA5T grid-cells

This use-case shows how a static dataset (in this case height/elevation at 250 m resolution) can be used to obtain information about other datasets such as ERA5T.

The goal here is to obtain the maximum difference in heights within a given ERA5T cell:

1. Select a given location or area.
2. Select the dataset GMTED250, height/elevation, surface, static.
3. Resample to domain ERA5T using linear interpolation to downscale and Minimum to upscale values.
4. In parallel, select again the same dataset and resample to domain ERA5T using linear interpolation to downscale and Maximum to upscale values.
5. Subtract the two values to get the difference in elevation.

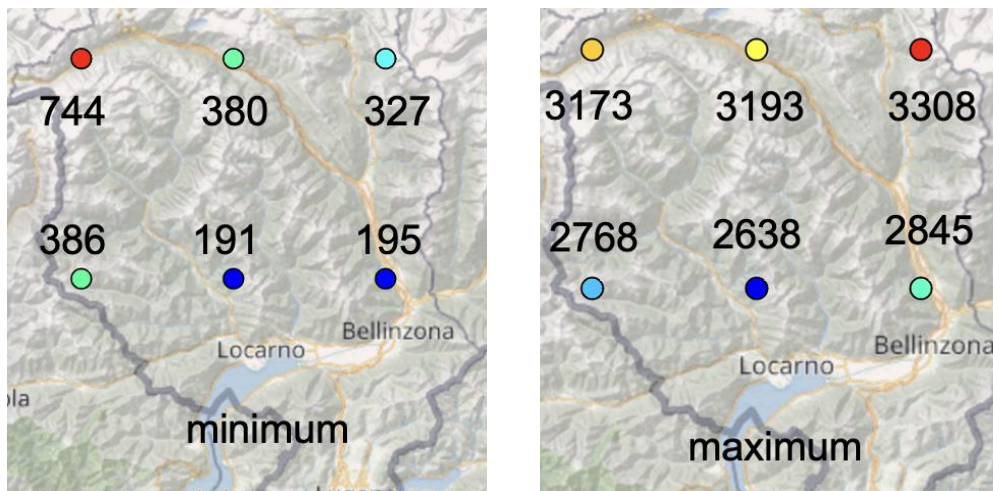


Figure 7: Minimum and maximum elevations (in meters) of ERA5T grid cells in Ticino.

3.3.2.4 Land-use of ERA5T grid cells

Using the static dataset GLOBCOVER (resolution 300 m), it is possible to select a given land cover type and see which is the percentage of it within an ERA5T grid-cell.

1. Select a given area.
2. Select the dataset GLOBCOVER, Land Cover Classification, Surface, Static.
3. Value is between **xxx** and **xxx** then pick **0** or **1**. (xxx is a general land cover type, such as 190, 11 or 20 – can find the list here: <https://docs.meteoblue.com/en/meteo/data-sources/datasets#globcover>)
4. Resample to domain ERA5T using Linear interpolation to downscale and mean to upscale.

3.4 eo-learn adapter

meteoblue offers access to environmental datasets like archived meteoblue weather simulations, ECMWF ERA5, gridded satellite information, soil properties and other reanalysis datasets. Terabytes of data are accessible through the meteoblue Dataset API.

Global weather models (NEMSGLOBAL or ERA5) cover the entire Earth. Some global models have been operational and calculated since decades, and provide consistent data sequences over 10-30 years, in hourly to 3-hourly intervals. NEMSGLOBAL and ERA5 are the only models available in hourly intervals for more than 30 years. The high-resolution weather (NEMS models are computed for single domains and therefore cover a certain region or continent and are not available globally like NEMSGLOBAL or ERA5. High resolution models (NEMS4, ICON7, etc.) are also only running since few years, and therefore their time series do not reach back for much longer than the year 2010. Most weather simulation datasets are automatically updated daily, which gives an instant availability of these datasets and underlines the consistency between weather forecast and history. The high-speed access through our dataset API allows to combine the advantages of the different datasets included.

The meteoblue Dataset API is a HTTP API and designed to offer a high degree of flexibility to retrieve and manipulate data for a lot of use-cases. A unique feature is the option process data directly on meteoblue servers to perform expensive transformations and retrieve only the relevant information from a dataset. The

concept of transformations is exploited to offer environmental data at satellite resolution in the Synergies EO-Learn platform.

To streamline the use of environmental and satellite data for machine learning, a special adapter to fuse meteoblue and Sinergise services was developed. The eo-learn meteoblue adapter offers direct access to the meteoblue Dataset API and prepares data to ingestion with machine learning routines in eo-learn.

The adapter was developed in cooperation with meteoblue and Sinergise and was released as open-source in the eo-learn GitHub repository.

The EOTasks for accessing weather data from meteoblue services are the eo-learn are available on GitHub: <https://github.com/sentinel-hub/eo-learn/blob/develop/io/eolearn/io/extra/meteoblue.py>.

3.4.1 Resampling to satellite resolution

Weather data and satellite data have notoriously different spatial resolutions: weather data resolutions range from 40 km to 2 km while Sentinel data is in the order of 60 m to 10 m resolution. Furthermore, weather models use specialised projections to efficiently compute orthogonal physical quantities or optimise storage requirements for global grids.

To facilitate experimenting with new ML methods as well as developing global, cross-scale, multi-purpose monitoring capabilities, weather data is spatially resampled to match satellite data. With a homogenous projection and resolution handling data from different weather models and performing calculations becomes significantly easier and faster.

Sentinel-2 satellite data commonly uses Universal Transverse Mercator (UTM) projection while weather data uses Arakawa, Gaussian or equirectangular projections. To efficiently resample data from weather models to UTM, analytical solutions for all projection and grid system have been implemented into the meteoblue dataset API. The API service can therefore efficiently resample to arbitrary satellite resolutions.

Two resampling techniques are implemented and can be selected depending on the scope of the analysis: nearest neighbour selection and bilinear interpolation.

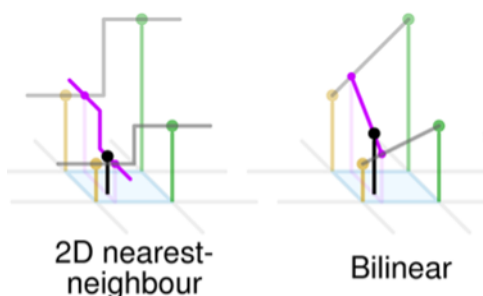


Figure 8: Comparison of nearest-neighbour selection and bilinear interpolation (Cmglee, CC BY-SA 4.0).

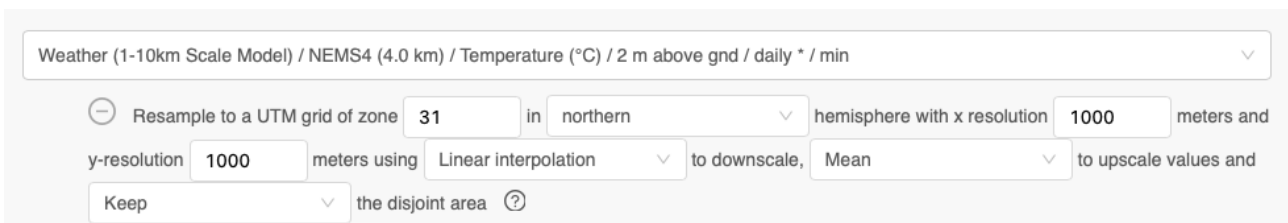
The nearest neighbour algorithm selects the weather variable value of the nearest point without considering the values of neighbouring points.

The bilinear interpolation algorithm computes instead, for each desired point, a weighted average of the meteorological attributes of the four surrounding datapoints (in diagonal directions) which is then applied as final interpolated value to the considered point.

If the desired resolution is lower than the weather model resolution (e.g., resample 2 km weather data to 10 km UTM), aggregation is performed. In the case of bilinear aggregation, the high-resolution values are distributed by weights to the aggregated dataset. This results in perfect inversion of bilinear interpolation.

Because of shape and projection of a weather models, interpolation and aggregation can occur in the same dataset. A weather model in equirectangular WGS84 projection with 0.1° resolution has 11 km resolution at the equator, but a resolution of 3 km at 70°N because the projection converges at the poles. The implemented resample algorithms perform aggregation and interpolation seamlessly and simultaneously.

Resampling to UTM projection has been integrated into meteoblue dataset APIs and web interfaces and is available through the protobuf outputs.



Weather (1-10km Scale Model) / NEMS4 (4.0 km) / Temperature (°C) / 2 m above gnd / daily * / min

Resample to a UTM grid of zone in hemisphere with x resolution meters and y-resolution meters using to downscale, to upscale values and the disjoint area

Figure 8: Resample to UTM projection in meteoblue dataset API.

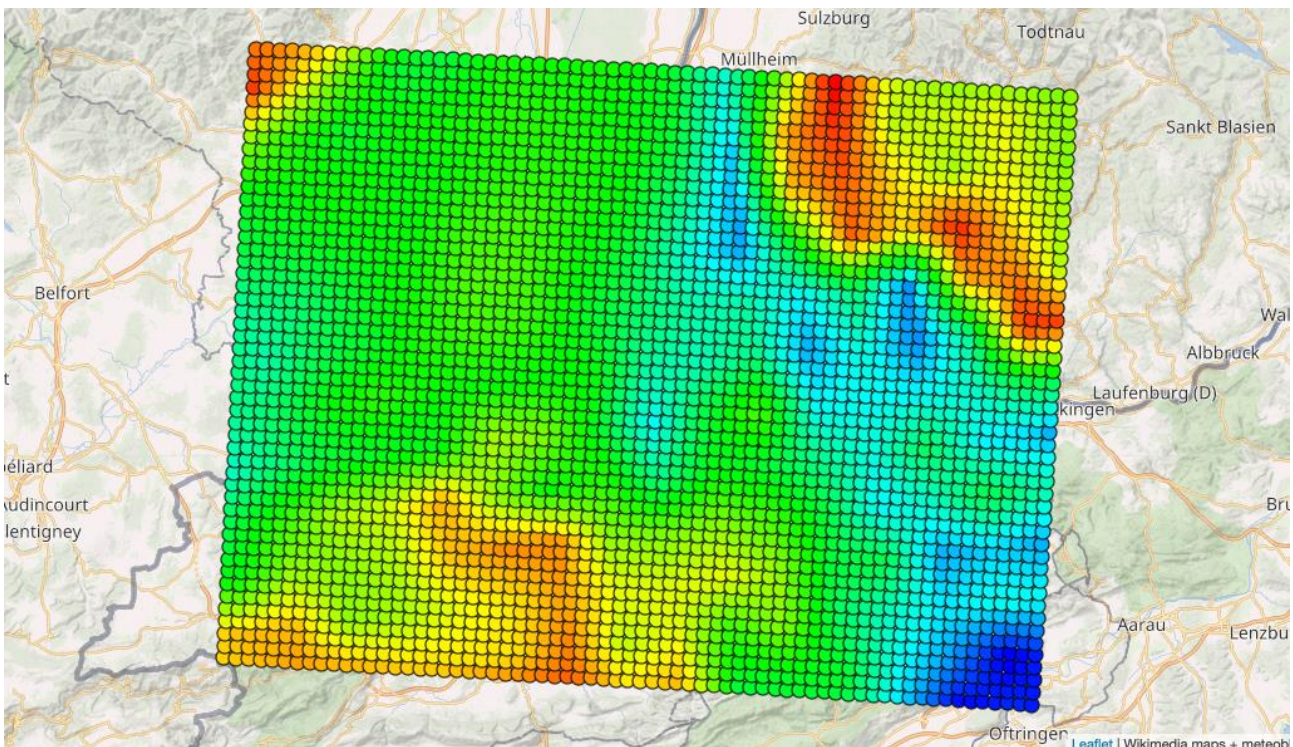


Figure 9: Resampled temperature to UTM satellite projection.

Additionally, temperature downscaling with elevation correction has been implemented into the service and works seamlessly with resampling to UTM satellite projection.

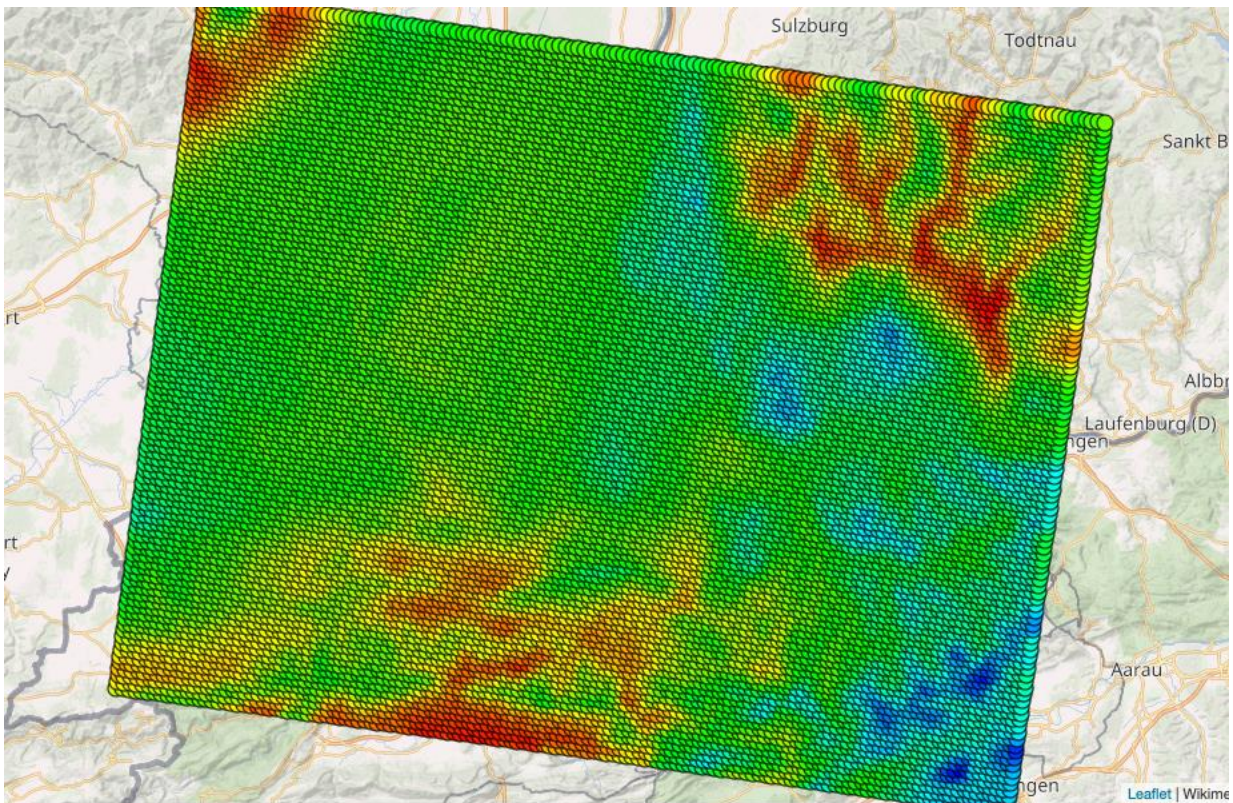


Figure 9: Resampling with temperature downscaling.

3.4.2 Raster and vector data tasks

To access all datasets in the meteoblue Dataset API, two separate EO-Learn EOTasks were implemented:

1. A vector task for arbitrary dataset projections and resolutions: [MeteoblueVectorTask](#)
2. A raster task for resampled data to satellite data projection: [MeteoblueRasterTask](#)

The vector task represents all coordinates of a dataset as a series of individual points. Therefore, weather data for individual coordinates as well as arbitrary areas or countries can be retrieved. Data is not aligned to satellite resolution but maintains the original resolution without any data modifications.

```

query = {
  "domain": "NEMS4",
  "gapFillDomain": None,
  "timeResolution": "hourly",
  "codes": [{"code": 11, "level": "2 m above gnd"}],
  "transformations": [
    {"type": "aggregateTimeInterval", "aggregation": "mean"},
    {
      "type": "resampleToUtm",
      "zone": 32,
      "hemisphere": "northern",
      "xResolutionMeters": 100,
      "yResolutionMeters": 100,
      "interpolationMethod": "linear",
      "spatialAggregation": "mean",
    }
  ]
}

```

Figure 10: Example of MeteoblueVectorTask

The code in Figure 10 shows an example of MeteoblueVectorTask, which will download NEMS4 data into a vector feature (tabular data with geolocation of each row, as shown in XX).

	TIMESTAMP	Longitude	Latitude	11_2 m above gnd_mean	geometry
0	2020-08-02 11:00:00	7.520231	47.497829	[22.458240509033203, 22.477575302124023, 22.49...	POINT (7.52023 47.49783)
1	2020-08-02 11:00:00	7.521558	47.497849	[22.458240509033203, 22.477575302124023, 22.49...	POINT (7.52156 47.49785)
2	2020-08-02 11:00:00	7.522885	47.497864	[22.458240509033203, 22.477575302124023, 22.49...	POINT (7.52289 47.49786)
3	2020-08-02 11:00:00	7.524213	47.497883	[22.458240509033203, 22.477575302124023, 22.49...	POINT (7.52421 47.49788)
4	2020-08-02 11:00:00	7.525540	47.497898	[22.458240509033203, 22.477575302124023, 22.49...	POINT (7.52554 47.49790)

Figure 11: Resulting weather (tabular) data from the MeteoblueVectorTask.

The raster task uses the data transformation capabilities of the meteoblue Dataset API. A spatial transformation to UTM satellite projection can be used to interpolate datasets to a matching resolution. Data can be retrieved and used as a two-dimensional raster-grid which fits satellite resolution perfectly. If the correct spatial extend and UTM zone of Sentinel-2 data is selected, weather data can be directly interpolated to align with Sentinal-2 datasets.

```

query = {
  "domain": "NEMS4",
  "gapFillDomain": None,
  "timeResolution": "hourly",
  "codes": [{"code": 11, "level": "2 m above gnd"}],
  "transformations": [
    {"type": "aggregateTimeInterval", "aggregation": "mean"},
    {
      "type": "resampleToUtm",
      "zone": 32,
      "hemisphere": "northern",
      "xResolutionMeters": 100,
      "yResolutionMeters": 100,
      "interpolationMethod": "linear",
      "spatialAggregation": "mean",
      "disjointArea": "keep"
    }
  ]
}

```

Figure 12: Example of MeteoblueRasterTask from eo-learn

The code in Figure 12 shows an example of MeteoblueRasterTask, which will download NEMS4 data in raster format. Combined with other eo-learn tasks (e.g., tasks to download EO data for that area), the code will produce EOPatch with both EO and weather/climate data, suitable for further analysis and processing.

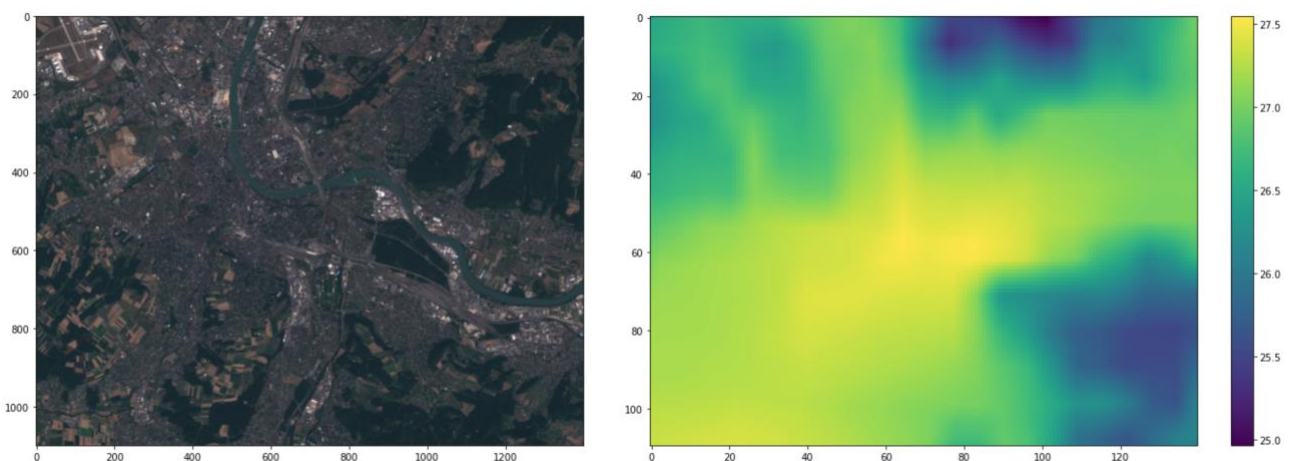


Figure 13: Single EO observation with corresponding weather data

Both vector and raster tasks were built to utilize the full potential of the meteoblue Dataset API. They support all transformations like spatial and temporal aggregations as well as transformations to detect drought periods, excessive rain, or successive heat days. For data scientists this enables an easy way to analyse climate change patterns in relation to satellite data.

3.5 Optimisation of meteoblue nowcasting and forecasting services

The meteoblue Learning Multimodel processing chain (Figure 14) has a number of extension points to deliver data via API. For example, the meteoblue Open Dataset SDK described in Section 3.1 provides access to the raw model and multi-model stages. The highest accuracy, and data most suitable to be used as ground truth, is available from the mLM stage. The optimisation has focused on this stage.

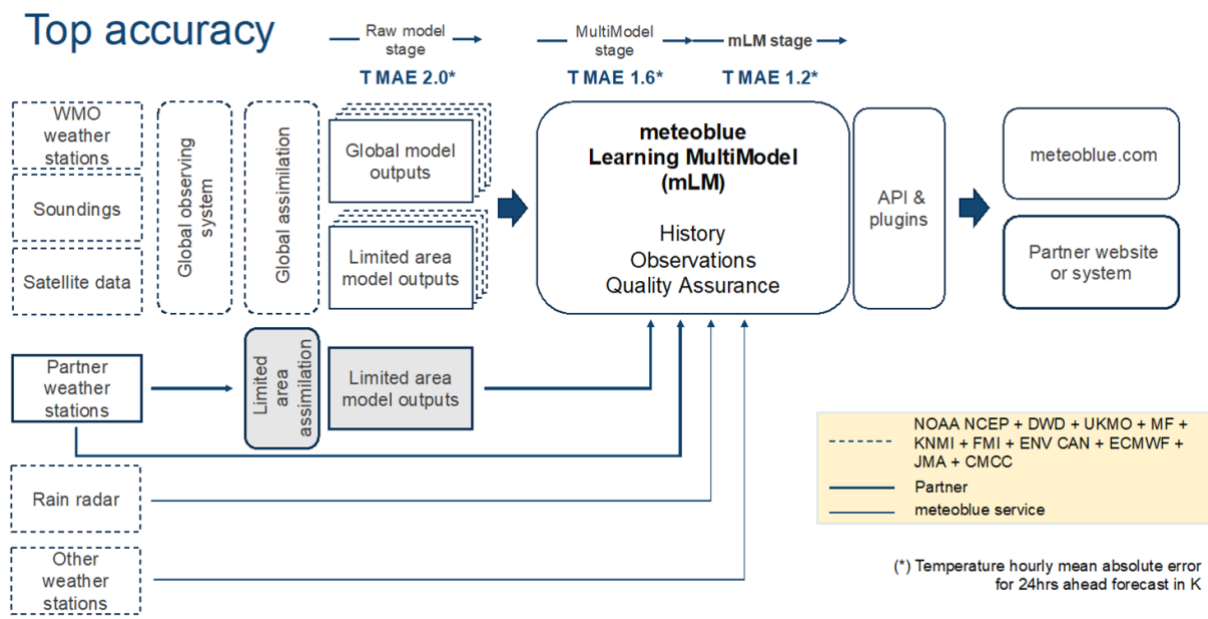


Figure 14: meteoblue Learning Multimodel process chain

The different data sources available to compute a weather forecast are updated at different times of day and at different intervals, ranging from twice a day for numerical weather prediction models to every 15 minutes for satellite images to every five minutes or less for precipitation radars and automatic weather stations. In the mLM stage a forecast is derived from all data available at the time the forecast is requested. This is particularly important for the nowcast (current weather situation and forecast for up to two hours) and hindcast (past weather situation, computed with weather information that has become available after the time of interest). Nowcasts and hindcasts have an accuracy that is closest to measurements with an actual station. This is used in "virtual weather stations"² that record nowcasts or hindcasts as if the data came from a weather station. Accuracy varies with data availability - it is higher in Central Europe than in Central Africa - but in general better than results from simply spatially interpolating weather station data.

² <https://docs.meteoblue.com/en/services/weather-stations/virtual-weather-stations>,
<https://www.agricolus.com/en/stazioni-meteo-virtuali-un-nuovo-servizio-allinterno-della-piattaforma-agricolus/>

3.5.1 The nowcasting pipeline

Within the scope of GEM project, meteoblue has significantly accelerated data extraction at the MLM stage to make using nowcast data as ground truth data practical. We describe this for the nowcasting subprocess.

Nowcasting is a complex process (Figure 15) ingesting always all available most up-to-date variables to improve the near-term forecast, nowcast, and hindcast.

meteoblue has significantly improved processing speed while decreasing resource usage of RAM and CPU, thus extending the practical range of use cases where weather data ground truth is critical in terms of variable and area covered. Of particular interest in this mix are accurate precipitation data derived from precipitation radar measurements.

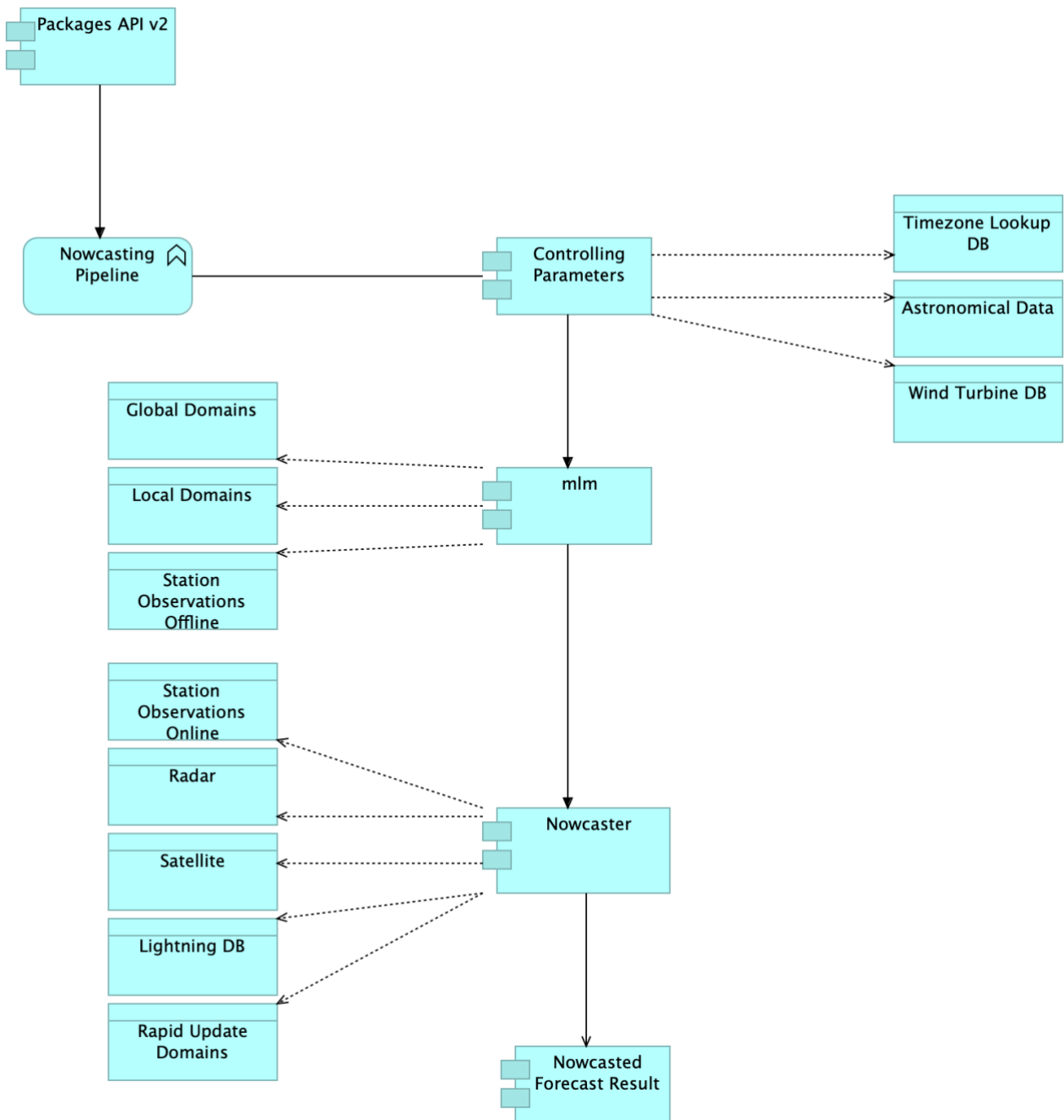


Figure 15: ArchiMate³ diagram of Nowcasting Pipeline within meteoblue API

3.5.2 Performance and reliability optimisations

The optimisations involved re-factoring the API code and rewriting it in a modern, type-safe, compiled language. Not shown in the diagram, the API is now a multi-threaded application running in parallel on multiple servers accessing data from optimised files on local SSD storage to meet performance requirements. The multi-

³ <https://www.opengroup.org/archimate-forum/archimate-overview>

threaded nature of the application posed some difficulties involving race conditions and occasional lock-outs that were analysed and removed.

Furthermore, the meteoblue new implementation greatly enhanced the unit testing coverage. With a comprehensive suite of tests, we are able to ensure a higher level of reliability and accuracy in the forecast data. The improved testing coverage not only boosts confidence in the API's performance but also minimizes the likelihood of potential bugs or data inconsistencies.

Another notable advantage of the overhauled package API is its cost-efficiency. By streamlining and optimizing the data retrieval and processing mechanisms, the API allows us to allocate resources more effectively, reducing unnecessary costs. The cost savings are particularly significant, especially in scenarios where large-scale weather data analysis is required.

In addition to performance and cost-efficiency, the adoption of a compiled language brings about better type safety, further enhancing the overall robustness of the system. The compiled nature of the language allows for early detection of potential errors and ensures that the forecast data delivered by the API is more reliable and accurate and can quickly be adopted to new developments in the field.

Table 1 shows a load test on one server (worker) with 10 CPUs (Intel Xeon CPU E5-2695 v3 @ 2.3 GHz) and 10 GB of RAM of the legacy API vs the optimised API. The load test is using Grafana k6⁴ to simulate a linear increase from 0 to 150 simultaneous users, followed by a linear decrease from 150 to 0 users, each during a time interval of 2 minutes. Each virtual user is sending a request which will return a nowcasted point forecast for a random location on Earth.

⁴ <https://k6.io/>

Table 1: Performance improvements from legacy system (IDL API) to optimised system (Swift API)

Metric	IDL API	Swift API
data_received	448 MB 1.9 MB/s	1.8 GB 7.5 MB/s
data_sent	4.8 MB 20 kB/s	15 MB 64 kB/s
http_req_blocked [median]	3µs	3µs
http_req_connecting [median]	0s	0s
http_req_duration [median]	496.16ms	47.53ms
http_req_duration with status ok [median]	1.09s	47.53ms
http_req_failed	38.09% ✓ 10093 ✗ 16399	0.00% ✓ 0 ✗ 83135
http_req_receiving [median]	4.78ms	1.27ms
http_req_sending [median]	17µs	17µs
http_req_tls_handshaking [median]	0s	0s
http_req_waiting [median]	489.49ms	45.47ms
http_reqs	26492 110.319153/s	83135 346.355287/s
iteration_duration [median]	496.56m	47.92ms
iterations	26492 110.319153/s	83135 346.355287/s
vus	1 min=1 max=149	1 min=1 max=150
vus_max	150 min=150 max=150	150 min=150 max=150
successful req/s	68	346

3.5.3 OpenAPI compliance

The API was adopted to be compliant with the OpenAPI⁵ standard in a backwards compatible way to make it more accessible for new users. Making the meteoblue forecast API comply with the OpenAPI standard facilitates collaboration and integration by providing standardized API documentation. It will be easier for users of the API to use automatic code generation tools, thus enhancing developer productivity, while built-in testing and validation ensure adherence to specifications. Additionally, the OpenAPI standard fosters a thriving ecosystem of shared APIs, encouraging innovation and collaboration. Overall, OpenAPI brings efficiency, interoperability, and a robust development framework to API implementation.

3.5.4 Python SDK (outlook)

Integrating the meteoblue forecast API into the meteoblue python SDK holds great potential for providing the additional benefits of accurate weather nowcast and forecast variables for a range of different applications. Users will be able to seamlessly switch between nowcast/forecasts and queries for historic data in their applications.

⁵ <https://www.openapis.org/>

3.6 High resolution temperature fields API

meteoblue has developed an API to retrieve hyper-resolution (10x10 m) temperature data. Examples in this section can be tried with a designated APIKEY, "5d2e1945322c-GEM-reviewers".

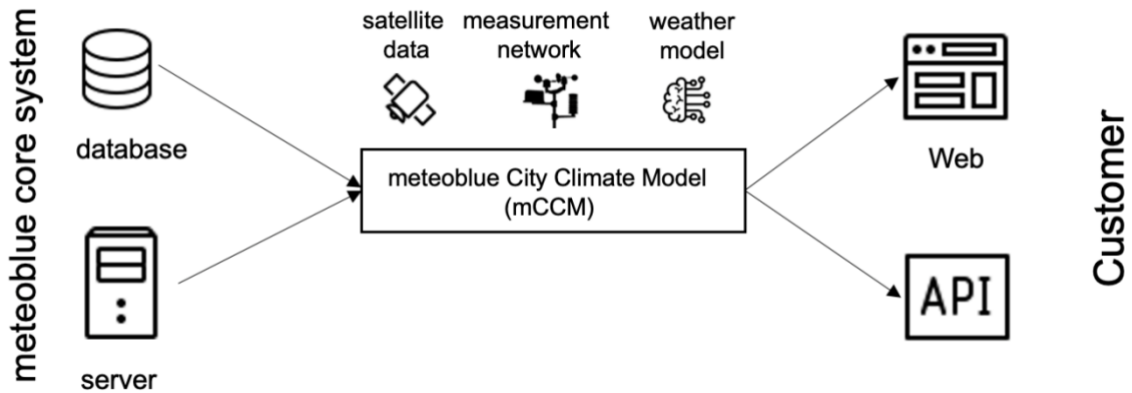


Figure 16: System architecture diagram of the integration of the meteoblue City Climate Model (mCCM) in the process chain of the meteoblue core system.

Figure 16 shows a system architecture diagram highlighting the meteoblue City Climate Model (mCCM) aspect of the meteoblue processing chain (Figure 14). It depicts how the data stream to the customers is ensured via a web interface or via the API, which is described in detail in the following sections.

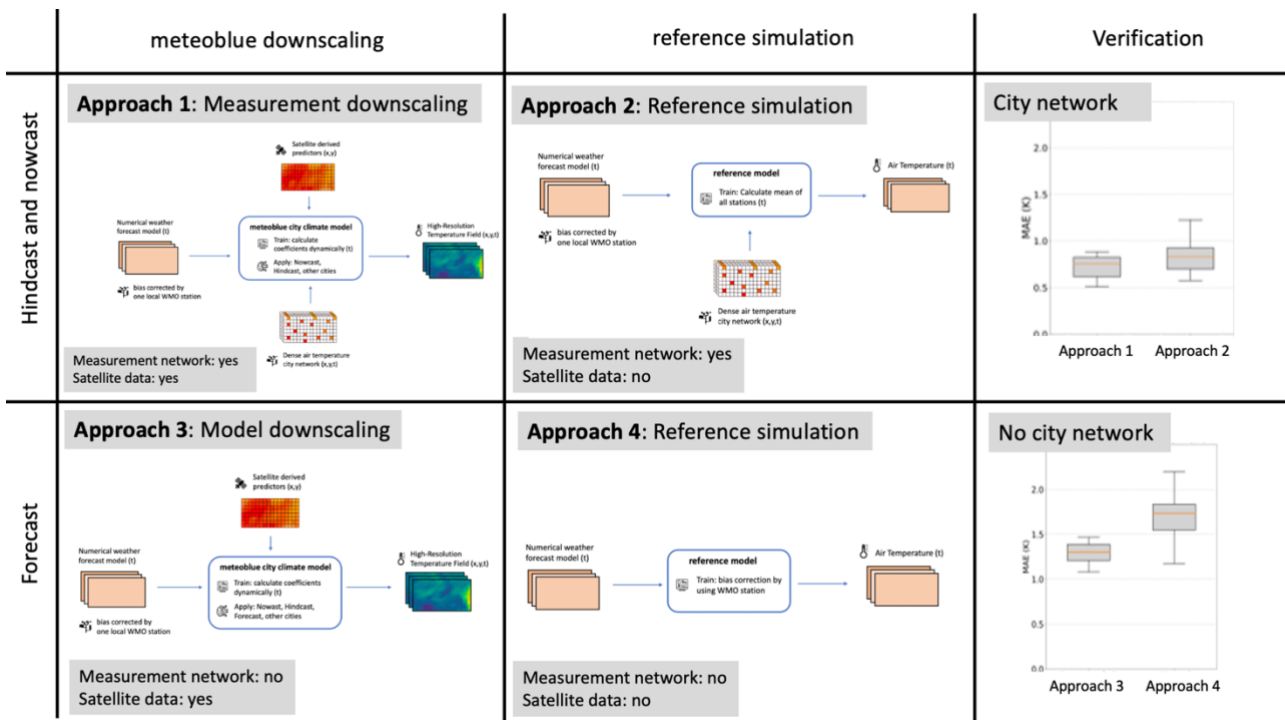


Figure 17: Components of the mCCM and their verifications.

Temperature downscaling in built-up areas is described in D2.3. A Jupyter Notebook⁶ documents the process. The methods of the approach are summarised in Figure 17. The data delivered via the High Resolution Temperature Fields API are based on “Approach 3” model downscaling. “Approach 1”, measurement-based downscaling, will be integrated in a later version of the API and apply where local measurement networks are available.

The web interface⁷ is a GEM demonstrator and forms part of the meteoblue freely accessible web site.

3.6.1 Data packages

The following four endpoints and data packages are available in the city climate API. This API resolves the small-scale variability of the air temperature field in a city and therefore can correctly model and forecast the urban heat island effect in cities.

- `/v1/city_domains/`
Available cities: number of cities available in the API including the extent of the city.
- `/v1/images/`
Images: JPGs of high-resolution air temperatures in 10 x 10 m by predefined colors and legends.
- `/v1/images_meta/`
Meta: Extent of image and meta information to reproduce color legend of JPG images.
- `/v1/data/`
Map data: Data of the high-resolution air temperatures in 10 x 10 m.
- `/v1/points/`
Point data: Timeseries of the air temperature for a set of points.

3.6.2 Available cities

The output of this endpoint is a list of cities available in the API and their extent, which is covered by the mCCM. Currently more than 80 cities are available.

Future⁸ example URL:

http://cityclimateapi.meteoblue.com/v1/city_domains?apikey=APIKEY

The following output data are available:

- city
- extent

⁶ https://gittext.sinergise.com/h2020_gem/code/temperature-downscaling-in-built-up-areas/-/blob/main/satellite%20pre-processing/satellite_pre-processing.ipynb

⁷ <https://www.meteoblue.com/en/products/cityclimate>

⁸ At the time of writing, this endpoint has not yet been made publicly available. This is expected for mid September 2023. The APIKEY for reviewers will be ‘5d2e1945322c-GEM-reviewers’. For questions please contact support@meteoblue.com.

Example JSON Output:

```
{
  "metadata":
    {
      "extent":
        "[
          lon_min,
          lat_min,
          lon_max,
          lat_max
        ]"
    },
  "units":
    {
      "latitude": "degree north",
      "longitude": "degree east"
    },
  "cities":
    [
      {
        "city": "adelaide",
        "extent":
          [
            138.4655,
            -35.05326,
            138.7404,
            -34.73142
          ]
      }
    ]
}
```

3.6.3 Images

The output of the “image” endpoint is a jpeg file, showing the air temperatures in a 10 x 10 m horizontal resolution for one specific timestep.

Example URL:

<http://cityclimateapi.meteoblue.com/v1/images?city=basel&time=current&apikey=APIKEY>

Input	Examples	Comments
city	berlin	Number of cities
timestamp	2023-07-13T07:00 current	YYYY-MM-DDThh:mm (in local time)
apikey	APIKEY	For accessing the API

The following output data are available:

- Image in jpg format

The variable “extent” returned by the “city_domains” endpoint (Section 3.6.2) defines the geographical extent for which an image is available for each city’s domain.

Images are available with a history of 72 hours. For the city Zurich a forecast image is additionally available 72 hours ahead. Note that the number of cities with a forecast will increase in the future.



Figure 18: Example of a heat map output for Berlin for 2023-07-13T07:00.

3.6.4 Images meta information

The output of the “image_meta” endpoint is a JSON file containing the geographical extent covered by the image and the required information to reproduce the colour legend for this image.

Example URL:

http://cityclimateapi.meteoblue.com/v1/images_meta?city=basel&time=current&apikey=APIKEY

Input	Examples	Comments
city	berlin	Number of cities
timestamp	2023-07-13T07:00 current	YYYY-MM-DDThh:mm (in local time)
apikey	APIKEY	For accessing the API

The following output data are available:

- Extent (city boundaries)
- Color legend (Hex color codes)
- Minimum and maximum of color legend

Example JSON Output:

```
{
  "city": "basel",
  "time": "2023-08-24 05:00:00",
  "extent": {
    "xmin": 7.45521,
    "xmax": 7.79286,
    "ymin": 47.43824,
    "ymax": 47.66685
  },
  "legend": {
    "color_palette": [
      "#433543", "#563853", "#5C3959", "#763E70",
      "#94438C", "#B967B1", "#AF72BD", "#AA96D7",
      "#A8B8E4", "#A7CDE3", "#95DBD7", "#77CCC6",
      "#76COCA", "#73ABC8", "#77A0D0", "#639CA7",
      "#5E9B90", "#6AA262", "#83A933", "#ABB033",
      "#CDB933", "#EDC233", "#FBAF33", "#F98A33",
      "#EE6933", "#B95233", "#974433", "#733933"
    ],
    "min": 15.64, "max": 23.64
  }
}
```

3.6.5 Map data

The output of the “data” endpoint is the high-resolution air temperature data for one timestep and a geographical extent (“latLonBox”), which needs to be defined by the user.

Input	Examples	Comments
city	berlin	Number of cities
timestamp	2023-07-13T07:00 current	YYYY-MM-DDThh:mm (in local time)
latLonBox	52.1,13.2,53.5,14.1	lat_min,lon_min,lat_max, lon_max
apikey	APIKEY	For accessing the API

Example URL:

<http://cityclimateapi.meteoblue.com/v1/data?city=basel&latLonBox=47.54,7.58,47.57,7.59&time=current&apikey=APIKEY>

Example JSON Output:

```
{
  "x": [7.5800,7.5801,7.5802,7.5803],
  "y": [47.5462,47.5463,47.5464,47.5465],
  "airTemperature": [17.41,17.88,18.12,18.11]
}
```

The variable “latLonBox” defines the geographical extent from which temperature data are extracted.

Temperature data are available with a history of 72 hours. For the city of Zurich forecast data is additionally available 72 hours ahead. Note that the number of cities with a forecast will increase in the future.

3.6.6 Point data

The output of the “points” endpoint is the high-resolution air temperature data for a temperature time series for a single point (to be extended in the future).

Input	Examples	Comments
city	berlin	Number of cities
lat	52.5058	WGS84 coordinates
lon	13.4164	WGS84 coordinates
start	2023-07-13T07:00 yesterday	Starting timestamp
end	2023-07-14T07:00 current	End timestamp
apikey	APIKEY	For accessing the API

Example URL:

<http://cityclimateapi.meteoblue.com/v1/points?city=basel&lon=7.58&lat=47.558&start=yesterday&end=current&apikey=APIKEY>

Example JSON Output:

```
{
  "coordinates":
    {
      "latitude":[47.558],
      "longitude":[7.58]
    },
  "time":
    ["2023-08-23 08:00:00","2023-08-23 09:00:00","2023-08-23 10:00:00"],
  "airTemperature":[
    [27.85,29.3,32.13]
  ]
}
```


Conclusions

This document provides an overview of meteoblue weather data and the APIs designed to access them, along with the descriptions of specific enhancements implemented to serve GEM specific needs, such as on-the-fly resampling to match satellite data resolution and gridded operations to seamlessly access data from different model domains.

New functionalities appositely designed for machine learning applications are documented and described in this report.

Of note is the development of a connector that allows eo-learn users to easily retrieve weather and climate data.

The deliverable constitutes moreover a reference documentation, where description and methodology to setup queries, retrieve data and perform operations are outlined.

The code to retrieve the data from meteoblue services through eo-learn has been made available on eo-learn repository: <https://github.com/sentinel-hub/eo-learn/blob/develop/io/eolearn/io/extra/meteoblue.py>.

The METEO/CLIMATE service is already supporting GEM's crop identification and conflict pre-warning map use-cases, and will be improved to respond to potential use-case specific needs during the upcoming task 2.9 "Optimization of METEO/CLIMATE services".