# MAchinE Learning for Scalable meTeoROlogy and climate

# First version of customised ML solutions

www.maelstrom-eurohpc.eu

# D1.3 First version of customised ML solutions

**Author(s):** Thomas Nipen (MetNor), Fabian Emmerich(4cast). Kristian Ehlert (4cast), Peter Dueben (ECMWF), Matthew Chantry (ECMWF), Nikoli Dryden (ETH), Ashkboos Salehn (ETH), Michael Langguth (FZJ), Bing Gong (FZJ), Yan Ji (FZJ)

**Dissemination Level:** Public

**Date:** 30/09/2022

**Version:** 1.0

**Contractual Delivery Date:** 30/09/2022

**Work Package/ Task:** WP1/ T1.3

**Document Owner:** ECMWF

**Contributors:** FZJ, ECMWF, MetNor, 4cast, ETH

**Status:** Delivered

ECMWF 4cast E4 COMPUTER ENGINEERING ETH zürich JÜLICH Forschungszentrum Norwegian Meteorological Institute uni.lu UNIVERSITÉ DU LUXEMBOURG

# MAELSTROM

# Machine Learning for Scalable Meteorology and Climate

**Research and Innovation Action (RIA)**
**H2020-JTI-EuroHPC-2019-1: Towards Extreme Scale Technologies and Applications**

**Project Coordinator:** Dr Peter Dueben (ECMWF)
**Project Start Date:** 01/04/2021
**Project Duration:** 36 months

**Published by the MAELSTROM Consortium**

**Contact:**
ECMWF, Shinfield Park, Reading, RG2 9AX, United Kingdom
Peter.Dueben@ecmwf.int

The MAELSTROM project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955513. The JU receives support from the European Union's Horizon 2020 research and innovation programme and United Kingdom, Germany, Italy, Luxembourg, Switzerland, Norway

# Contents



# Table

## Figure

# 1 Executive Summary

This document provides information on the Tier-2 large datasets for the six MAELSTROM applications (AP1-AP6) that are published in the scope of this deliverable. The machine learning solutions are further developed and upgraded. A series of experiments are performed, and the corresponding scientific discoveries for each application are documented in this report.

The increase in complexity of the machine learning solutions in comparison to the configurations of Deliverable 1.1 is obvious as two of the applications now perform distributed deep learning training with multiple GPUs to accelerate the training process. This report evaluates the performances in terms of the model accuracy and speed-up.

# 2 Introduction

## 2.1 About MAELSTROM

To develop Europe's computer architecture of the future, MAELSTROM will co-design bespoke compute system designs for optimal application performance and energy efficiency, a software framework to optimise usability and training efficiency for machine learning at scale, and large-scale machine learning applications for the domain of weather and climate science.

The MAELSTROM compute system designs will benchmark the applications across a range of computing systems regarding energy consumption, time-to-solution, numerical precision and solution accuracy. Customised compute systems will be designed that are optimised for application needs to strengthen Europe's high-performance computing portfolio and to pull recent hardware developments, driven by general machine learning applications, toward needs of weather and climate applications.

The MAELSTROM software framework will enable scientists to apply and compare machine learning tools and libraries efficiently across a wide range of computer systems. A user interface will link application developers with computer system designers, and automated benchmarking and error detection of machine learning solutions will be performed during the development phase. Tools will be published as open source.

The MAELSTROM machine learning applications will cover all important components of the workflow of weather and climate predictions including the processing of observations, the assimilation of observations to generate initial and reference conditions, model simulations, as well as post-processing of model data and the development of forecast products. For each application, benchmark datasets with up to 10 terabytes of data will be published online for training and machine learning tool-developments at the scale of the fastest supercomputers in the world. MAELSTROM machine learning solutions will serve as a blueprint for a wide range of machine learning applications on supercomputers in the future.

## 2.2 Scope of this deliverable

### 2.2.1 Objectives of this deliverable

To publish the Tier 2 large dataset per application. To upgrade machine learning solutions and test them on with the Tier 2 dataset. To perform various experiments on different machine learning solutions to understand their functionality for the applications. To perform distributed deep learning to accelerate the Machine Learning (ML) training. To compare machine learning frameworks (PyTorch and TensorFlow). To benchmark with various numerical precision.

### 2.2.2 Work performed in this deliverable

Deliverable 1.3 focus on the scientific discovery of the six applications by applying a large dataset (Tier 2 dataset) and using advanced machine learning solutions. The description of the Tier 2 dataset is provided, which is built upon the Tier 1 dataset in Deliverable 1.1. Various machine learning solutions from Deliverable 1.2 are enhanced and applied to Tier 2 dataset. The performances of the ML solution for each application are further improved, and scientific discoveries are made and presented in this report.

While the large datasets and advanced ML solutions are delivered, the distributed machine learning through data parallelism is established for two applications (AP3 and AP4). The results will be relevant for the hardware analysis which will take place in WP3.

In this deliverable, various levels of numerical precision are tested for AP3 in machine learning in weather and climate modelling to understand how the numerical format can influence the training performance. We also compare the performance of the two main machine learning frameworks, TensorFlow and PyTorch, for AP 5.

### 2.2.3 Deviations and counter measures

There are no significant deviations from the planned contributions of the deliverable.

# 3 Advanced Machine Learning solutions for the Six applications

## 3.1 AP1: Blend citizen observations and numerical weather forecast

The goal AP1 is to produce high resolution (1x1 km) hourly temperature forecasts 58 hours in advance for the Nordic countries.

### 3.1.1 Dataset

The dataset is as described in Deliverable D1.1, with a few notable changes. Firstly, the dataset now includes 7 extra predictors (marked by * below).

| Predictor name | Unit | Has leadtime? |
|---|---|---|
| **2m temperature control member** | °C | Yes |
| **2m temperature 10% from ensemble*** | °C | Yes |
| **2m temperature 90% from ensemble*** | °C | Yes |
| **Cloud area fraction** | 1 | Yes |
| **Precipitation amount** | mm | Yes |
| **10m x wind component** | m/s | Yes |
| **10m y wind component** | m/s | Yes |
| **2m temperature bias previous day** | °C | Yes |
| **2m temperature bias at initialization time** | °C | No |
| **True altitude*** | m | No |
| **Model altitude*** | m | No |
| **True land area fraction*** | 1 | No |
| **Model land area fraction*** | 1 | No |
| **Standard deviation of analysis at initialization time*** | °C | No |

*Table 1: Tier 2 dataset of AP1 for temperature and precipitation forecast*

Secondly, the dataset's time period has been reduced from 5 to 2 years. The MEPS modelling system changes frequently over time as model physics, data assimilation, and ensemble perturbation methods are improved. We chose the 2-year period 2020/03/01 to 2022/02/28, which represents a time period where the model configuration remained relatively constant. Some dates in the dataset are missing due to missing data in MET Norway's archive. The dataset includes Numerical Weather Prediction (NWP) runs that were available at 03Z, 09Z, 15Z, and 21Z. Including all forecast initialization times for every day of the two-year period would yield too large a dataset. Instead, only one of the 4 runs is included for each day, where the provided runtime alternates from day to day. Including data from the 4 runs is important as it forces the ML solution to deliver forecasts that work for any forecast initialization time. Data is stored in 675 netCDF files, one for each forecast run.

### 3.1.2 Experimental setup

The prediction problem is to predict three percentiles: 10%, 50%, 90%. This means the output layer in the ML-models have 3 predictands. This application uses the quantile score as a loss function, as described in Deliverable D1.1.

| Metric | Tier 2 dataset |
|---|---|
| **Data size** | 6.2 TB |
| **Grid size** | 2321x1796 |
| **Spatial resolution** | 1 km |
| **Lead times** | 58 (0, 1, 2, … 58) |
| **Time period** | 2020/03-2022/02 |
| **Number of predictors** | 14 |
| **Predictor size (time, leadtime, y, x, predictor)** | 675 * 58 * 2321 * 1796 * 14 |
| **Target size (time, leadtime, y, x)** | 675 * 2321 * 1796 * 14 |

*Table 2: Summary of Tier 2 dataset for AP1*

The models are trained using the first 50% of the dataset (2020/03/01 to 2021/02/28). Each experiment is trained for 1 epoch, making one pass over the training data. The training reads the dates of the NWP runs in a random order. As a single training sample is large (i.e. only 365 samples in total), the domain was split into 63 patches each of a size of 256x256 points to increase the number of parameter updates in an epoch. This results in a total of 22,806 training batches and parameter updates for one epoch.

A small validation dataset was used to record the quality of the solution as the training progresses. A 29GB subset of the total dataset was used, where 24 dates were selected that samples all months of the year and all forecast initialization times. Additionally, the validation used only a subset of the full domain, including part of the North sea, southern Norway, and southern Sweden. The validation grid has size of 512 x 1024, with its southwest corner at x-index 300 and y-index 550. This was done to keep the validation dataset small enough to fit in main memory while at the same time including a wide variety of weather situations. Validation takes 42s and is run every time 4 input files have been processed. This results in a ~12% performance penalty, but allows the testing stage to select the model parameters that are best.

The model parameters that had the best validation loss during the training were saved and used for testing. The test dataset consists of the second half of the full dataset (2021/03/01 to 2022/02/28).

Each predictor was normalised by subtracting its mean and dividing by its standard deviation. The same normalisation coefficients were used for all grid points, times, and lead times.

### 3.1.3 Benchmark models

Four benchmark models are tested. The simplest is a linear regression (LINREG) model of the 14 predictors. This is implemented as a one-layer neural network with a linear activation function between the inputs and outputs.

The second model is a dense neural network (DNN), where five dense layers, each with five units, are connected by a rectified linear unit (ReLu) activation function. These layers are connected to the three outputs by linear activation functions. Weights are shared across grid points and lead times.

The third model is based on 2D convolutional neural network (CNN) layers. The basic architecture consists of three convolutional layers, each with a 3x3 stencil. The layers have 12, 5, and 5 filters respectively. A dense layer connects the convolutional layers and the outputs. ReLU is used as the activation between layers, and linear activation is used before the output layer. Weights are shared among all lead times.

The fourth model is a U-Net model (Ronneberger et al., 2015) consisting of 3 levels, with 16, 32, and 64 output channels. The model weights are shared among all lead times. The convolution size is 3x3 and the up and downsampling ratio is 2.

Additionally, the models are compared against the raw model output (RAW) that has not been post-processed.

### 3.1.4 Experiment 1: Comparing benchmark models

In this experiment, we compared the performance of RAW, LINREG, DNN, CNN, and U-Net (see Figure 1 and Table 3 below). All models improve the forecast accuracy over RAW by 6-11%. CNN performed similarly to the simpler LINREG and DNN networks, suggesting that the added complexity isn't able to extract complex relationships between the input and output. The more complex UNET model, however, performed better than the other models. We did not analyse the reason for the performance increase, but we suspect it is related to the model's ability to detect large scale features.

### 3.1.5 Experiment 2: Comparing leadtime-dependent weights vs. leadtime as a predictor

The prediction task produces forecasts for all 59 lead times simultaneously. We know that some predictors (e.g., bias_recent) are more relevant to early lead times than later ones. To introduce lead time information into the models, we compare adding lead time as a (static) predictor field and adding a locally-connected layer at the end where each lead time has different parameters. We tested this on the best model from experiment 1 (U-Net).

Adding lead time as a predictor led to a small improvement of scores for lead times 0-6h, but no noticeable improvement later on. The lead time layer significantly improved scores for lead time 0, but reduced the quality for all lead times after that. The overall modest improvements from lead time information was surprising to us, but may indicate that lead time information can be inferred by the model (e.g. the spread of the ensemble is a proxy for lead time).

### 3.1.6 Experiment 3: Adding new features

In this experiment, new features were added to the input. These include the x and y grid values, the month of the year, and the hour of the day. We expect these predictors will aid the model in differentiating seasonal, diurnal, and spatially varying biases. We tested this on the model that performed best in experiments 1 and 2 (U-Net). We did not, however, find a noticeable difference in quality when introducing these features.

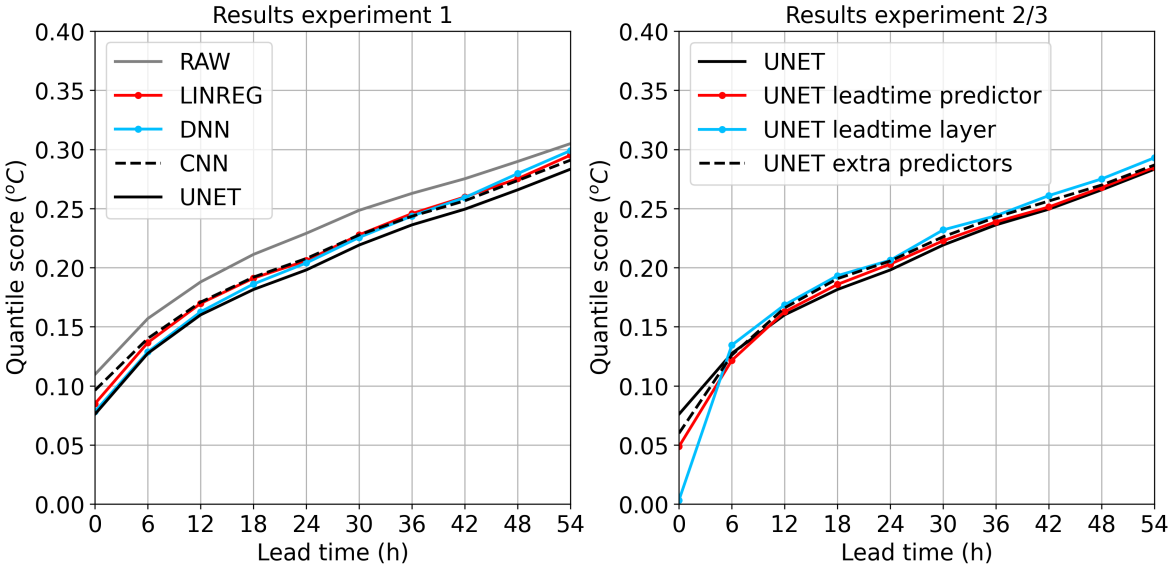


*Figure 1 Test scores for the different experiments as a function of forecast lead time.*

| Model | Parameters | Test loss | Improvement |
|---|---|---|---|
| Experiment 1 | | | |
| **RAW** | 0 | 0.2249 °C | 0 % |
| **LINREG** | 45 | 0.2094 °C | 6.89 % |
| **DNN** | 183 | 0.2067 °C | 8.09% |
| **CNN** | 2,437 | 0.2091 °C | 7.03% |
| **UNET** | 115,893 | 0.1995 °C | 11.29% |
| Experiment 2 | | | |
| **UNET leadtime predictor** | 116,040 | 0.1986 °C | 11.69 % |
| **UNET leadtime layer** | 119,895 | 0.2043 °C | 9.16% |
| Experiment 3 | | | |
| **UNET leadtime layer and extra predictors** | 116,628 | 0.2032 °C | 9.65% |

*Table 3: Summary of test scores for all experiments in AP1*

### 3.1.7 Outlook

We will investigate several approaches to further improve the predictive accuracy of the models and the computational performance of the training pipeline.

**Improving predictive accuracy**

The experiments showed that the U-Net is a promising model capable of beating simpler models. We still believe that the model can be further tuned to make better use of the input predictors. For example, adding extra features in experiment 3 did not lead to improved predictions. We will investigate if the network structure can be altered in a way that better finds the signals that potentially exist in this dataset. The NWP model used in the dataset has known biases that we hope the ML solution would be able to identify. We will investigate if these are indeed detected, and experiment with different network structures that are better able to detect these biases.

We will also properly tune the many hyper-parameters of the U-Net, such as number of levels and output channels. Since the dataset has a very high resolution (1x1km), it is possible that the 3 levels in the U-Net is not sufficient to allow large-scale meteorological features to be recognized.

**Improving computational performance**

In the experiments, the data loader streams data from the disk to the GPUs. As the model is trained, the next batch of data is fetched and processed in parallel. The processing pipeline for the most expensive model (U-Net, in experiment 3) looks like this:

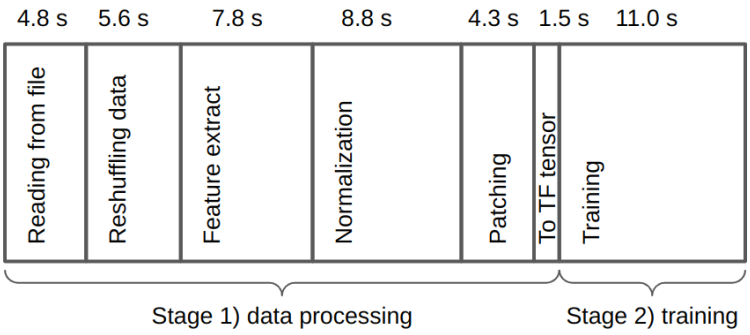


*Figure 2:Diagram for typical processing and training times for one file (10GB) when training a UNET model on an A100 GPU on Jewels-Booster. The two stages are run in parallel in a pipeline fashion. Since stage 2 is currently faster than stage 1, the pipeline is bound by the data processing step. This is also consistent with the analysis in the Deliverable 3.4, indicating that I/0 is still a main bottleneck for the computing performance of ML pipeline. Additionally, validation is performed after every 4th file and takes 29 seconds.*

The data processing steps are done on-the-fly to allow the ML modeller the flexibility of experimenting with different data processing settings. This, however, makes the pipeline bound to the speed of the data processing, and not the training. Hence, GPU utilisation is far from optimal. We plan to alleviate

this problem by parallelizing the code that performs the various data processing steps. We anticipate that with some work, the pipeline will eventually be bound by the training.

We will also investigate distributed training, by exploiting data parallelism.

### 3.1.8 Data and Code access

The code to run the experiments is found here[1]. Installing the python package in the repository provides the command-line tool maelstrom-train, which starts a run based on configuration defined in YAML files. This includes options for the data loader, models, training, optimization, loss, and output. The exact configuration of each experiment is documented in the settings found in etc/exp1.yml, etc/exp2.yml, and etc/exp3.yml.

| Experiment # | Run command |
|---|---|
| **1** | maelstrom-train --config etc/raw.yml --hardware gpu<br>maelstrom-train --config etc/exp1.yml --hardware gpu |
| **2** | maelstrom-train --config etc/exp2.yml --hardware gpu |
| **3** | maelstrom-train --config etc/exp3.yml --hardware gpu |

*Table 4: Run commands for the Experiments in AP1.*

## 3.2 AP2: Incorporate social media data into the prediction framework

AP 2 aims to use data provided by social media as weather sensors. As a first step, we use texts of Tweets to predict the occurrence of rain ("raining", "not raining") at the Tweet's location and time of creation.

### 3.2.1 Datasets

We use historical tweets from the years 2017-2020 that include keywords related to the presence of rain, e.g., "rain", "sunny" or "drizzle." Locations need to be attached to these tweets such that we can assign weather data to them. Locations can be assigned to Tweets in two ways. The user may opt in to be tracked by Twitter such that GPS data are assigned to every tweet of the location of the user. Alternatively, users can tag their tweets from a curated list of locations by Twitter. We use Tweets with locations from either method. However, we exclude tweets that are tagged with a location that surface area is greater than 100 square kilometres, which corresponds to the spatial resolution of our weather data. Here, we focus on tweets in the English language that originate from any location within the United Kingdom. This allows us to resort to the most popular pretrained models in natural language processing (NLP), which are usually trained in English. In total, our dataset includes 1.3 million Tweets, 1 million of which have a tagged location and 300k have GPS coordinates attached. 700k Tweets are tagged "not raining" while 600k Tweets are tagged "raining".

[1] https://github.com/metno/maelstrom-train

Precipitation data are taken from the ECMWF-IFS weather forecasting model during the years of 2017-2020. The model has a spatial resolution of 0.1° in both horizontal and vertical directions and a temporal resolution of 1h. Precipitation above the noise level of the data is interpreted as "raining" by our model. We assign Tweets precipitation values based on the nearest data point in both space and time. We plan to study the impact of any form of averaging for this assignment in the future.

To improve the results, texts are preprocessed to match vocabulary of the model and the input as well as focusing the model's attention on the relevant parts of the text. We remove the hyperlinks and hashtags (unless they represent a keyword) and rephrase colloquial terms. If emojis are directly related to precipitation, e.g. umbrella, rain cloud …, these emojis are converted to text. Remaining emojis are removed from the text. For training, we only retain texts that still contain at least one keyword after preprocessing. In addition, we remove all Tweets that contain the word "Sun" (note, capital "S"), as this word appears to be mostly used when referring to the British newspaper "The Sun" or referring to Sunday, which means that these Tweets usually contain insufficient information for our prediction task.

For Experiment 1, we only use a subset of our dataset, which corresponds to all Tweets posted in 2020. This allows us to assess the relevance of training set size. For the second Experiment, we use our full dataset. We use 80% of our datasets for training and withhold 20% for testing.

### 3.2.2 Metrics and loss function

We use cross entropy as our loss function. The proficiency of the model is evaluated via the f1 score of the minority class ("not raining"). The f1 score is the harmonic mean of the precision and recall.

### 3.2.3 Overview models

Generally, our problem can be phrased as a text classification task, where deep learning models based on the transformer architecture (Vaswani et al., 2017) achieve state-of-the-art results (e.g. Yang et al., 2019). Transformers use self-attention to attribute varying relevance to different parts of the text. Based on transformers the BERT model was developed (Devlin et al., 2019), which includes crucial pre-training steps to familiarise the model with relevant vocabulary and semantics. In addition, the DeBERTa model (He et al., 2021a) adds additional pre-training steps and disentangles how positional and text information is stored in the model to improve performance. For this experiment we rely on the most recent version of the model, which is DeBERTaV3 (He et al., 2021b).

We use the default version of DeBERTaV3_small as described in He et al., 2021b, with an adopted head for text classification. For this, we add a dropout layer with user-specified dropout rate and a pooling layer that passes the embedding of the special initial token ([CLS], comprising the meaning of the whole Tweet) to the loss function.

### 3.2.4 Experiment 1 - Small dataset

We initialise the model with pre-trained weights as explained in He et al., 2021b. We then employ a lower learning rate compared to the pre-training step to train the full model with our small dataset, which only contains Tweets posted in 2020. For both our experiments, we use default parameters as

given in He et al., 2021b. For hyper-parameters, we use parameters as listed in their Table 10. After hyper-parameter tuning, we only deviate in the following parameters that we use for both our experiments, where our batch size is 32, learning rate is 3e-5, drop out of the task layer is 0.1, weight decay is 0.01, training epochs is 1 and the warm up ratio is 0.45. Our best model achieves a f1 score of the minority class ("raining") of 0.64.

To analyse our results, we use the relative difference between the outputs of the final softmax layer to the respective label as a measure of confidence of the model. First, we focus on tweets labelled as "raining", where the model predicts "not raining" at a high confidence.

### 3.2.5 Experiment 2 - Large dataset

Analogously to Experiment 1, we initialise our weights with pre-trained weights from He et al., 2021b and train with parameters described in Experiment 1. Our best model achieves a f1 score of the minority class of 0.66.

The Figure 3 summarises our results in the form of a normalised confusion matrix. The confusion matrix in the left-panel corresponds to our 2020 dataset (Experiment 1) and the right-hand panel corresponds to the whole dataset.

### 3.2.6 Results and Outlook

Our best model achieves an f1 score of the minority class ("raining") of 0.66 when including all Tweets. The smaller dataset (only Tweets from 2020) achieves a somewhat lower f1 score of the minority class ("raining") of 0.64. The datasets yield an AUC (Area under the ROC Curve) of 0.77 and 0.76, respectively.

In the future, we aim to use larger model variants and experiment with alternative deep learning models, e.g. XLNet (Yang et al., 2019) or ULMFiT (Howard et al., 2018). In addition, exploring alternative ML techniques like XGBoost (Chen et al., 2016) or the more recent CatBoost (Prokhorenkova et al.,2018) which also allows the user to include embeddings for training.

### 3.2.7 Data and code access

The data is available via the S3 buckets at ECMWF and can be downloaded using the CliMetLab `climetlab-maelstrom-social-media` plugin[2,3]. We directly provide the precipitation data used in this study. In compliance with Twitter terms of service restrictions the Tweets will not be shared in a public repository instead the Tweet IDs are available in the repository. The code of the application is provided on the 4Cast GitHub account and instructions for usage are provided.

[2] https://github.com/4castRenewables/climetlab-plugin-a2

[3] https://pypi.org/project/climetlab-maelstrom-social-media

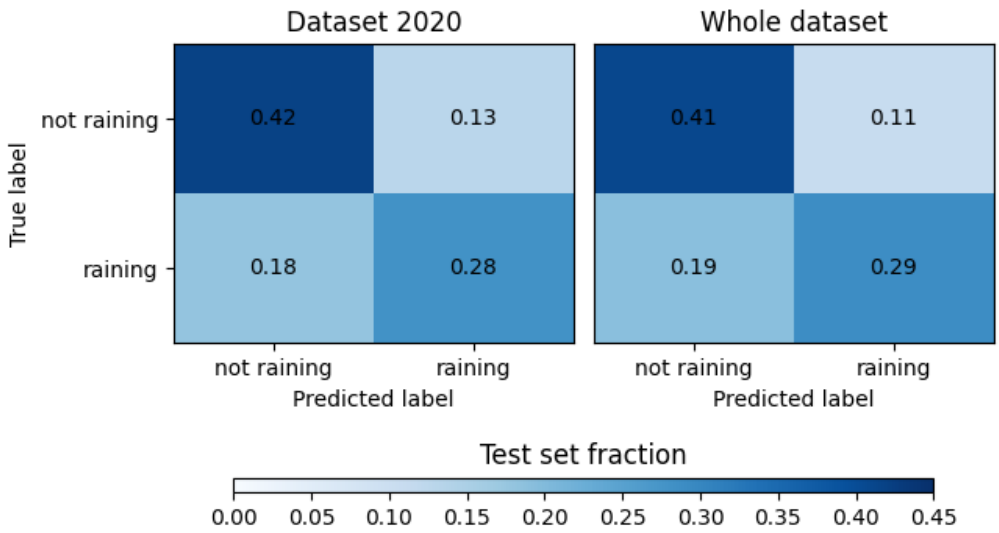


*Figure 3: Confusion matrix of experiments results for AP2*

## 3.3 AP3: Build neural network emulators to speed-up weather forecast models and data assimilation

### 3.3.1 Dataset

Application 3 seeks to emulate the radiative transfer process both for radiation with short and long wavelengths. This is a columnar problem, i.e. sets of inputs & outputs are formed of atmospheric columns, and the task is found within all weather and climate models. The inputs are profiles of atmospheric state and composition, e.g. temperature and aerosol properties. The inputs either live on model levels (137 levels in the IFS model which is used for the study), model half-levels (138 levels), model interfaces (136 levels) or as scalars. A complete list of the variables can be found in deliverable 1.1 or in the CliMetLab plugin. The CliMetLab provides the inputs gathered by level type, such that 3 input arrays are provided, containing the half-level, full-level & surface variables.

For the outputs, there is a duality between down and up fluxes, the typical output of a conventional scheme, and heating rates. At the surface, fluxes are necessary to heat and cool the surface, in the atmosphere, the heating rates are the desired variable that is used to increment temperature profiles. The two variables are related through

$$\frac{dT}{dt} = \frac{g}{c_p} \frac{F^n_{i+1/2} - F^n_{i-1/2}}{p_{i+i-1/2} - p_{i+i-1/2}}$$

where the left-hand side denotes the heating rate, $F^n$ denotes the net (down minus up) flux, p the pressure, g is the gravitational constant, $c_p$ is the specific heat at constant pressure of moist air and $i$ is the vertical index. We will throughout the experiment encode this relationship in a neural network layer and dually learn to minimise errors of the heating rate and flux profiles. Our metrics of interest are therefore the RMSE and MAE of the fluxes and heating rates.

For this phase of reporting, we introduce an extended version of the dataset. This extends the number of example columns used to train the models for learning the radiative transfer process. These can be accessed from the `maelstrom-radiation-tf` CliMetLab dataset using the `subset = "tier-2"` argument. This generates a dataset with 21,640,960 examples. This is to be contrasted with the Tier 1 dataset which contained only 67,840 examples. We also introduce Tier 2 subsets for validation and testing, accessible with `"tier-2-val"` and `"tier-2-test"`. Each of these contain 407,040 columns from 2019 (training data are taken from 2020). This dataset requires ~400Gb of disk space, a significant amount. We therefore also introduce Tier 3 datasets, which are equivalent for the validation & test splits, but contain "only" 2,984,960 columns, consuming ~60Gb of disk space. We found significant model improvements when using the Tier 2 dataset over the Tier 3 dataset, even when controlling for the number of iterations. Here, we will only present results on the Tier 2 data.

For brevity, we focus our reporting on the shortwave heating process. Equivalent models are being trained for longwave heating, with the current leading architecture proving to be optimal for both wavelengths.

### 3.3.2 Experiment 1 – Convolutional backbone

The first network we consider features a convolutional structure. Before this can be applied, the input shapes need to be standardised. Surface variables are repeated 138 times to match the 138 half-levels. Full-level fields are zero-padded at the top of the atmosphere by 1 to match 138. Interface fields are zero-padded at both the top & bottom to match 138. All inputs are concatenated along the channel dimension, resulting in 47 input channels. The learnt component of this model consists of convolutional layers, each with 64 filters, a kernel width of 5 and the Swish nonlinearity (Ramachandran, 2017). First, we use 5 convolutional layers featuring dilated convolutions (Chen et al.,2017), with dilation factors [1,2,4,8,16]. These are used to propagate information throughout the vertical column. Following these we use 5 more convolutional layers without dilation. The next step is to include a final convolutional layer with 2 channels, representing the downwards and upwards fluxes, with a sigmoid activation. This output is multiplied by two scalars, the global incoming solar radiation and the cosine solar zenith angle. This scales the output in proportion to the local incoming flux and provides our output fluxes. Finally, we use a custom layer to enact the heating rate equation outlined in 3.3.1. This takes the fluxes and half-level pressure as inputs. The graph of this model is depicted below (Figure 4).

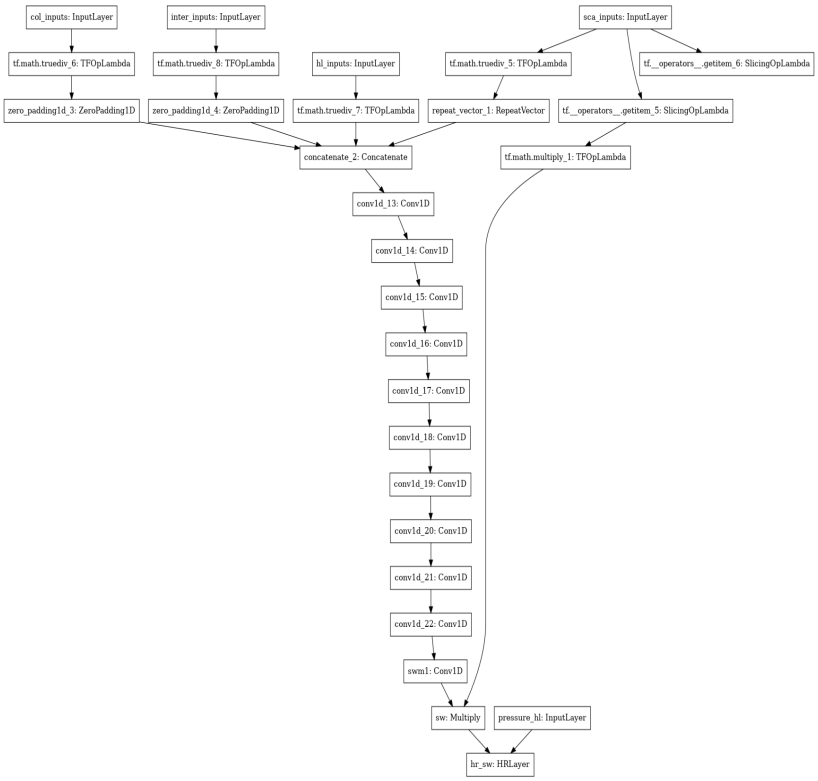


*Figure 4: The model structure of Experiment 1 for AP3*

We train the model for 50 epochs with a batch size of 512 and an initial learning rate of $2x10^{-4}$. The learning rate is multiplied by a factor of 0.25 after 4 epochs without validation loss improvement, with a minimal learning rate of $10^{-6}$. Our loss function is a combination of the mean-absolution-error (MAE) of the normalised flux profiles and the MAE of the heating rate. The normalised flux profile is calculated by dividing the flux profile by the incoming downwards flux. This standardises each column to be 1 for downwards flux at the top of the atmosphere and means that atmospheric columns with acute solar zenith angles, i.e. dusk/dawn are weighted equally with midday profiles. To combine the normalised MAE of the flux profile and the MAE of the heating rate we multiply the latter by 0.1, empirically finding that this led to a good balance of priorities. We select the best model on the validation dataset as the optimal model.

### 3.3.3 Experiment 2 – Recurrent neural network backbone

The second design follows that of Ukkonen (2021), who aimed to predict a similar process but for a shallower atmosphere. The design is motivated by the algorithm used in the conventional solver, which integrates down the atmospheric column, calculates interactions with the surface, integrates back up the column, and finally back down the column one final time. To mimic this, the design of Ukkonen uses a series of recurrent neural networks for each of these passes. Specifically, the network begins with a Gated Recurrent Unit (GRU, Cho et al.,2014.), which integrates from the top of the atmosphere to the surface, i.e. 137 steps of the GRU, where the appropriate full-layer variables are inputted at each step down the atmosphere. The output of the GRU at the surface is combined with

the surface albedo, before applying a Dense layer to the output to mimic the reflectance at the surface. A second GRU is then integrated from the surface to the top of the atmosphere, where the layerwise inputs are the inputs and outputs from the previous GRU layer. A third GRU integrates back down the column, with the inputs and outputs from both prior GRU layers. Each of the GRU layers use 64 units, tanh activation and sigmoid recurrent activation. Finally, we reproduce the last features of the convolutional model outlined in experiment 1; a convolutional layer with 2 channels, multiplication by incoming flux and a heating rate layer. This model, without these final steps, is depicted in Figure 4 of Ukkonen (2021), and a representation of the graph is shown in Figure 5. In total, the model has 85,698 trainable parameters.

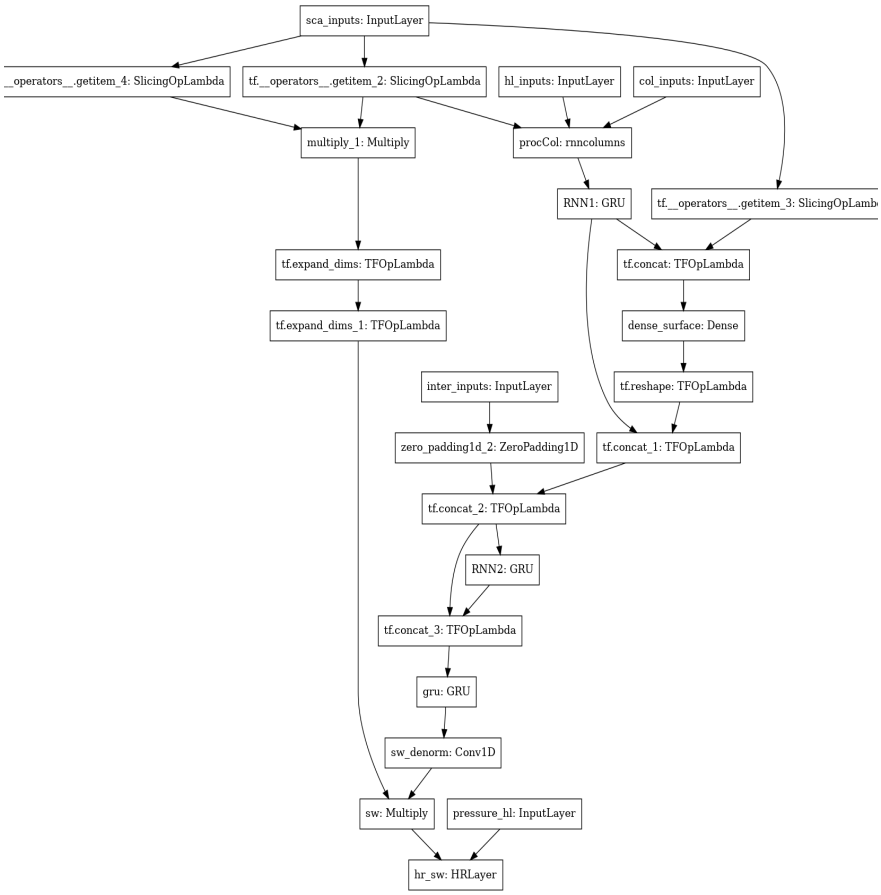


*Figure 5: The model structure of Experiment 2 for AP3*

Training follows that of the convolutional model, except the initial learning rate is set to 0.001.

### 3.3.4 Experiment 3 – Convolutions augmented with self-attention

The third experiment is an extension of the convolutional model in experiment 1. The only changes introduced are two self-attention layers (Vaswani 2017), which are introduced after the dilated convolution and before the final convolution layer. These self-attention layers have 8 heads and a key dimension of size 8. The purpose of introducing the self-attention layers is to aid the propagation of

information throughout the vertical column. As these layers enable rapid exchange of data in the vertical dimension we remove the dilation from the early convolution layers. In total, the model has 233,922 trainable parameters. The graph of the network is depicted in Figure 6.

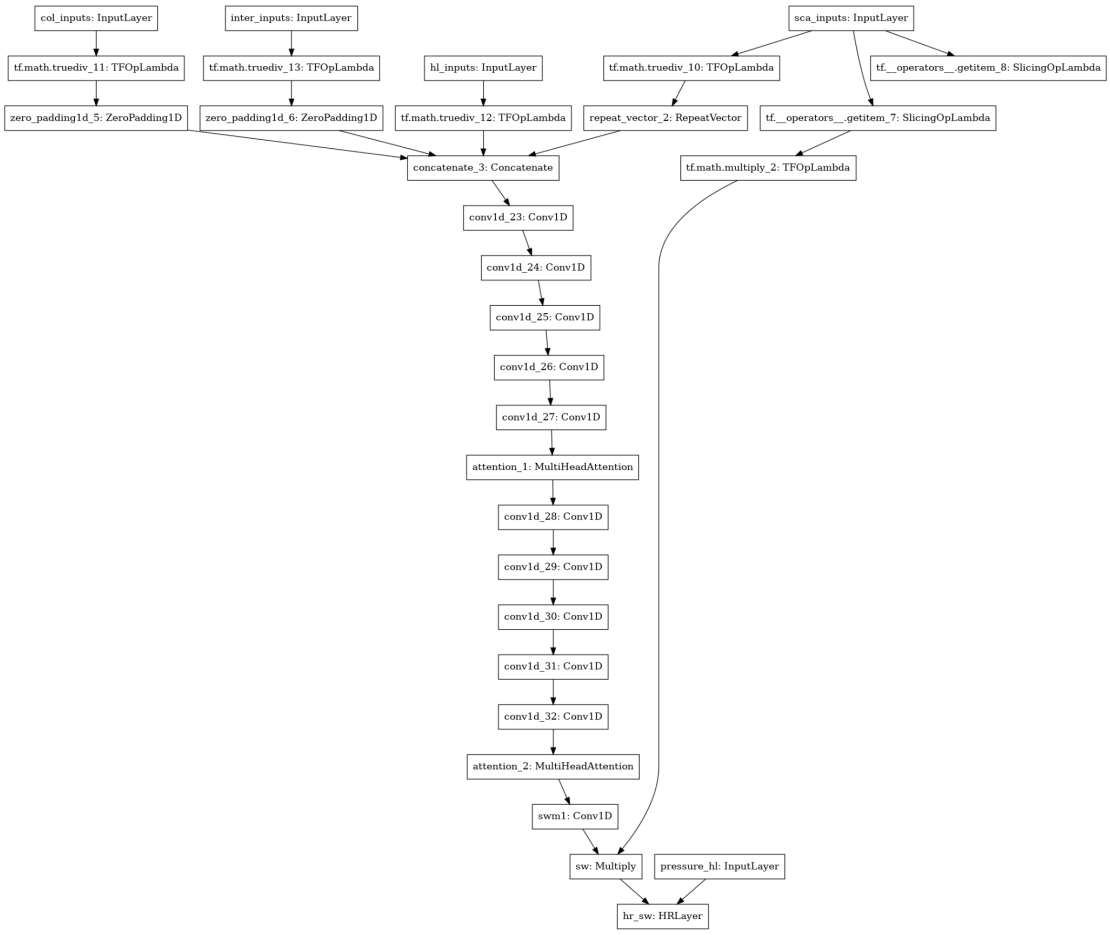


*Figure 6: The model structure of Experiment 3 for AP3*

During testing, we found that the default behaviour on NVIDIA A100s with Tensorflow to use Tensor Float 32 led to worse scores. Therefore this behaviour was removed for the training of this model, resulting in single precision being used for all calculations

### 3.3.5 Results

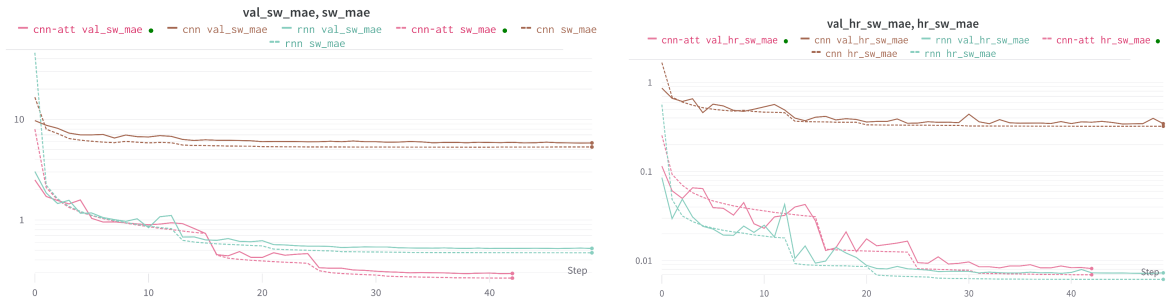


*Figure 7: The training and validation performance for the three experiments. X-axis indicates the number of training steps; Y-axis is the loss values.*

Figure 7 shows the training and validation curves for the three experiments, with the left plot showing the results for the fluxes and the right plot the heating rates. For the fluxes, experiment 3 produces the best results on both the training and validation dataset. For the heating rates, the RNN from experiment 2 produces the best results, but only by a small margin.

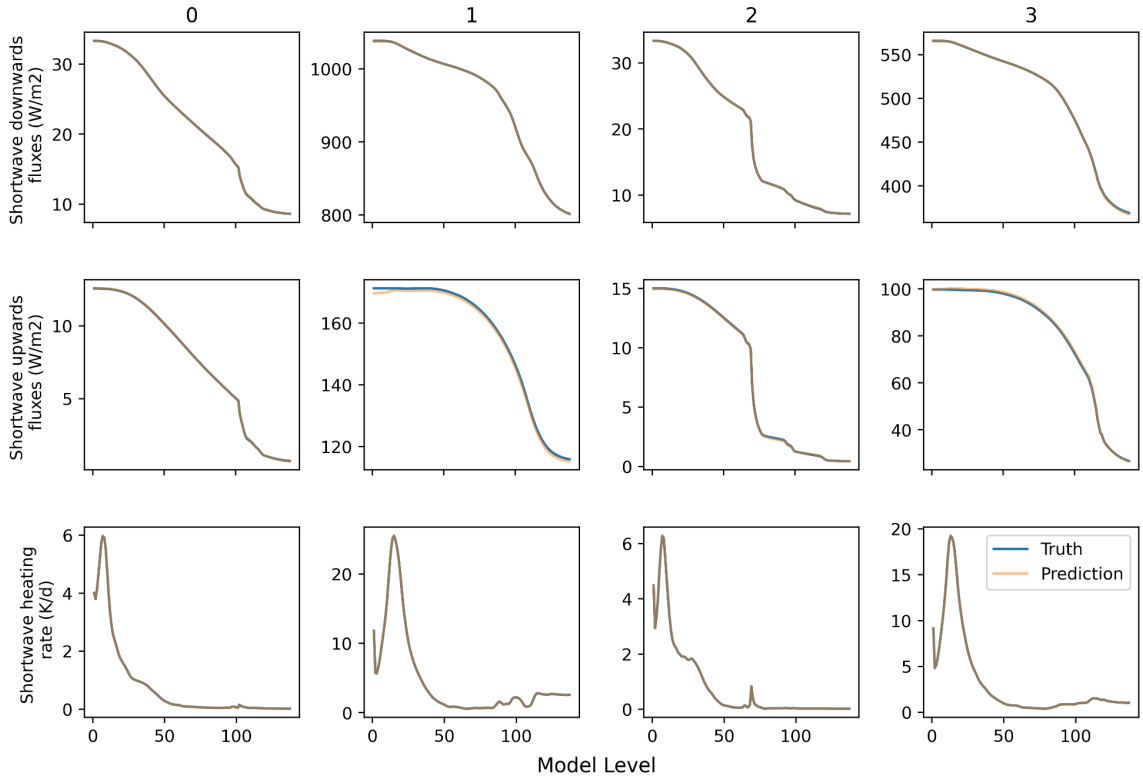


*Figure 8: Four example columns from the test data for the shortwave fluxes (down & up) and the heating rate*

The above figure plots four example columns from the test data for the shortwave fluxes (down & up) and the heating rate. The x-axis corresponds to the vertical dimension, with model level 1 corresponding to the top of the atmosphere. The RNN from experiment 2 is depicted as the

prediction. Overall, extremely good agreement is seen, with only a small degradation in the upwards flux at the top of the atmosphere for example 1.

| Experiment # | Loss | SW MSE | SW MAE | SW HR MSE | SW HR MAE |
|---|---|---|---|---|---|
| **1** | 0.0299 | 1627 | 12.2600 | 0.0963 | 0.0763 |
| **2** | 0.0017 | 1.6180 | 0.4290 | 0.0004 | 0.0061 |
| **3** | 0.0014 | 0.5000 | 0.2498 | 0.0005 | 0.0070 |

*Table 5 : The performance on the test data in Figure 8*

On the test data the same picture is observed, the results are displayed in Table 5. Both experiments 2 & 3 can be considered optimal depending on the importance of the two components.

### 3.3.6 Outlook

The final test of any model for this application is coupling it into the IFS atmospheric model to assess forecast quality. This step will be carried out next to differentiate between the two models. The relative inference costs will also be an important factor if the scientific quality of the forecasts are comparable for both models. A method for introducing these models into the Fortran codebase has been developed[4]. but this has not yet been tested using GPUs for the inference.

Further model optimisation is surely possible, as some of the hyperparameters in the above models have not been fully explored.

These and other architectures will be examined for the longwave aspect of the heating problem.

Finally, the models for this application will have further value, providing differentiable models for data assimilation. Over the next year we will establish the best methodology for generating the tangent linear versions of our models, and then testing their stability within data assimilation.

### 3.3.7 Data and code access

Data and code are both provided through the climetlab-maelstrom-radiation repository/pip package. Installation can be done with

```
pip install climetlab-maelstrom-radiation
```

Once installed, shell commands to run each of the experiments can be found in the Table 6 below. Note that the first time one of the experiments is run a data download will occur. With a good internet connection this will take ~3 hours and require ~400Gb of disk space. To run the code with a minimal data download (~3Gb), use `--tier 1` below, while noting that the model performance will be dramatically reduced. Details on the location of the data have been documents[5].

[4] https://github.com/ecmwf-projects/infero
[5] https://climetlab.readthedocs.io/en/0.9.9/guide/caching.html

| Experiment # | Run command |
|---|---|
| **1** | radiation-benchmarks-sw --epochs 50 --batch 512 --model cnn --tier 2 |
| **2** | radiation-benchmarks-sw --epochs 50 --batch 512 --model rnn --tier 2 |
| **3** | radiation-benchmarks-sw --epochs 50 --batch 512 --model cnn --tier 2 --attention --notf32 |

*Table 6: Run commands for the Experiments in AP3*

## 3.4 AP4: Improve ensemble predictions in forecast post-processing

Weather prediction needs to not only forecast the most likely future scenario, but also provide a probability distribution over specific weather events. Traditionally, existing numerical weather prediction systems achieve this by generating many ensemble members, or perturbed realisations of a weather simulation, which are then used to estimate probability distributions for key quantities, such as precipitation. However, achieving high-quality distributions requires many ensemble members, which is computationally expensive.

To address this challenge, Application 4 utilises deep neural networks to post-process these ensemble members to improve the quality of the probability distribution. As a first step, we have developed a new dataset, ENS-10, and an associated set of ML models, to serve as a benchmark for the ensemble post-processing task. This dataset is also documented in Ashkboos et al., 2022, which provides additional details.

### 3.4.1 Dataset

The ENS-10 dataset consists of ten perturbed ensemble members spanning twenty years (1998–2017) for medium-range weather forecasts (lead time 48 hours). This data was introduced in MAELSTROM Deliverable 1.1, and we summarize key details here. Each ensemble member contains a selected set of key weather variables over eleven pressure levels (10, 50, 100, 200, 300, 400, 500, 700, 850, 925, and 1000 hPa):

- U wind component,
- V wind component,
- Geopotential,
- Temperature,
- Specific humidity,
- Vertical velocity,
- and Divergence.

The dataset also contains important surface variables:

- Sea surface temperature,
- Total column water,
- Total column water vapour,

- Convective precipitation,
- Mean sea level pressure,
- Total cloud cover,
- 10 m U wind component,
- 10 m V wind component,
- 2 m temperature,
- Total precipitation,
- and Skin temperature at the surface.

ENS-10 is generated from reforecasts run at ECMWF, with two forecasts per week. The data are provided on a structured grid at 0.5° resolution.

### 3.4.2 Tasks and Benchmarks

As part of ENS-10, we define the prediction correction task, where the goal is to correct the output distribution of a set of ensemble weather forecasts. Formally, given a set of ensemble members at time $T$ and lead time $L$, the task is to predict a corrected cumulative distribution function (CDF) for a variable of interest (e.g., temperature) at time $T + L$. In ENS-10, we consider three variables for this task: 2 m temperature (T2m), temperature at 850 hPa (T850), and geopotential at 500 hPa (Z500). Years 1998–2015 are used as training data, and years 2016–2017 are used as a test set. The ERA-5 reanalysis dataset produced by ECMWF is used to provide ground-truth values.

A potential solution is evaluated with two different metrics. The first is the standard continuous-ranked probability score (CRPS) (Zamo et al., 2018). CRPS generalises the mean absolute error for the case where forecasts are probabilistic. In addition, in Ashkboos et al., 2022, we introduce the *Extreme Event Weighted CRPS* (EECRPS) metric to evaluate a solution specifically for extreme events, which are of particular interest in probabilistic forecasting. EECRPS is the pointwise CRPS weighted by the Extreme Forecast Index (EFI), which measures the deviation of the ensemble forecast relative to a probabilistic climate model.

We provide three baseline ML methods: Ensemble Model Output Statistics, a multi-layer perceptron, and a U-Net.

### 3.4.3 Experiment 1 – Ensemble Model Output Statistics (EMOS)

Ensemble Model Output Statistics (EMOS) (Gneiting et al., 2005) is a statistical method based on a linear relation between the ensemble values and the corrected mean and variance. It computes the mean and standard deviation of the corrected distribution by fitting a linear function to the ensemble members.

### 3.4.4 Experiment 2 – Multi-Layer Perceptron (MLP) Network

A multi-layer perceptron (MLP) is a set of full-connected layers, each followed by a non-linearity (e.g., ReLU). We apply an MLP with one hidden layer to ENS-10, which estimates the pointwise corrected mean and standard deviation of the output distribution. The network has 25 and 17 input dimensions

for the volumetric and surface variables, respectively (two inputs for each variable in the dataset and three for XYZ coordinates). The hidden layer has dimension 128 and uses a ReLU activation.

### 3.4.3 Experiment 3 – U-Net Network

The MLP model operates on each grid point separately. We also apply a U-Net model (Ronneberger et al., 2015), which is widely used in image segmentation and has previously been used for ensemble post-processing (Grönquist et al., 2021).

Our U-Net baseline consists of three levels, each with a set of [convolution, batch normalisation, ReLU] modules and operates on the whole grid with 22 and 14 input dimensions for the surface and volumetric variables, respectively (two inputs for each variable). We use convolutions with 32, 64, and 128 output channels in the network's first, second, and third levels, respectively.

### 3.4.4 Results

We evaluate our benchmark models on the ENS-10 dataset through verification metrics CRPS and EECRPS, and additionally provide the raw ensemble's mean and standard deviation with no correction ("Raw" in the results). Data were normalised for input. The U-Net was trained using a batch size of eight samples, while the EMOS and MLP models used one sample per batch (larger batch sizes degraded accuracy). All models were trained for ten epochs on a single A100 GPU, which took 0.75, 0.25, and 1 hours for the EMOS, MLP, and U-Net, respectively.

Table 7 shows the final results of our models, trained using either ten (10-ENS) or five (5-ENS) models. We report the mean and standard deviation over three experiments for all results (except Raw, where this is not applicable).

| Metric | Model | **Z500** [m2s-2] | | **T850** [K] | | **T2m** [K] | |
|---|---|---|---|---|---|---|---|
| | | 5-ENS | 10-ENS | 5-ENS | 10-ENS | 5-ENS | 10-ENS |
| **CRPS** | Raw | 81.030 | 78.240 | 0.748 | 0.719 | 0.758 | 0.733 |
| | EMOS | 79.080±0.739 | 81.740±6.131 | 0.725±0.002 | 0.756±0.052 | 0.718±0.003 | 0.749±0.054 |
| | MLP | 75.840±0.016 | 74.630±0.029 | 0.701±2e-4 | 0.684±4e-4 | 0.684±6e-4 | 0.672±5e-4 |
| | U-Net | 76.660±0.470 | 76.250±0.106 | 0.687±0.003 | 0.669±0.009 | 0.659±0.005 | 0.644±0.006 |
| **EECRPS** | Raw | 29.80 | 28.78 | 0.256 | 0.246 | 0.258 | 0.250 |
| | EMOS | 29.100±0.187 | 30.130±2.166 | 0.248±3e-4 | 0.259±0.018 | 0.245±0.001 | 0.255±0.018 |
| | MLP | 27.860±0.006 | 27.410±0.010 | 0.240±1e-4 | 0.234±2e-4 | 0.233±2e-4 | 0.229±2e-4 |
| | U-Net | 27.980±0.240 | 27.610±0.490 | 0.235±0.003 | 0.230±0.002 | 0.223±5e-4 | 0.219±0.001 |

*Table 7: Global mean CRPS and EECRPS on ENS-10 test set for our baseline models*

### 3.4.5 Data and Code Access

Data may be accessed via a CliMetLab plugin, installed as:

```
pip install climetlab climetlab-maelstrom-ens10
```

The ENS-10 dataset can then be downloaded using

```
import climetlab as cml
cml.load_dataset('maelstrom-ens10')
```

Full details on experiments and additional examples can be found at github[6].

## 3.5 AP5: Improve local weather predictions in forecast post-processing

AP5 explores the application of deep neural networks for statistical downscaling of meteorological fields. Inspired by the success of super-resolution techniques on images in the computer vision domain, different model architectures will be explored for downscaling the 2m temperature over complex terrain. These architectures, along with the deployed dataset, are described subsequently.

### 3.5.1 Dataset

In deliverable 1.1, a dataset for a pure downscaling task has been published. Pure downscaling in this context means that the deep neural network is only learning a mapping from the coarse-grained input data to the high-resolved target data. Systematic biases due to the truncated spatial resolution of the input data are not part of the downscaling task, even though their reduction constitutes one of the central aims when increasing the spatial resolution of atmospheric models (see, e.g., Giorgi and Gutowski et al., 2015).

In this deliverable, we aim to extend the statistical downscaling task by creating a dataset that allows to incorporate both, increasing the spatial resolution of coarse-grained meteorological fields to high-resolved field including bias correction. For this, we choose the ERA5-reanalysis as the input and the COSMO-REA6 reanalysis as the target dataset.

The ERA5-dataset constitutes a global reanalysis product which is based on ECMWF's Integrated Forecasting System (IFS) model, version CY41R2 (Hersbach et al., 2021). Data is available from 1959 until near real-time so that the spatio-temporal coverage is comprehensive. However, the underlying model runs on a reduced Gaussian grid with a spacing of about 31 km ($\Delta x_{ERA5} \simeq 0.2825°$) which is fairly coarse-grained.

By contrast, the COSMO-REA6 dataset is provided on a rotated pole grid with a spacing of about 6 km ($\Delta x_{CREA6,rot} = 0.055°$, see Bollmann et al., 2015 a,b). Thus, the spatial resolution is about five times higher than the (global) ERA5-reanalysis product. Anyway, the COSMO-REA6 is limited to a domain covering Europe (880x856 grid points) and the reanalysis data is only available between 1995 and August 2019. The COSMO model (version 4.25) developed at the German Weather Service (DWD) was used to create the regional reanalysis dataset.

For the Tier-2 dataset of this deliverable, we harmonise data from 2006 to 2018 over a target domain with 120x96 grid points covering large parts of Central Europe. Although the target domain is just a subset of the COSMO-REA6 domain, it constitutes a suitable region for the downscaling task due to the underlying complex topography of the German low-mountain range and parts of the Alps (see

[6] https://github.com/spcl/ens10

Figure 9). To avoid mismatches in the underlying grid projection of both datasets, the ERA5-reanalysis data is remapped onto a rotated pole grid with $\Delta x_{ERA5,rot} = 0.275°$. The same remapping procedure as described in deliverable 1.1 is deployed.

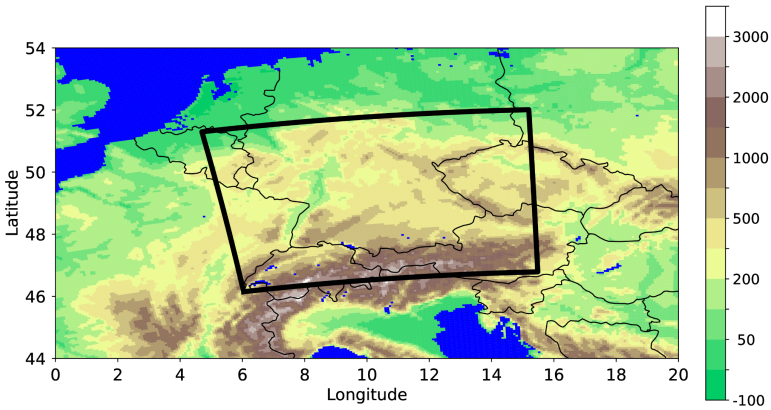


*Figure 9: Surface topography in metre above sea level from the COSMO REA6-dataset. The target domain of the ERA5 to COSMO-REA6 downscaling task is rendered in black. The domain comprises 120x96 grid points in zonal and meridional direction on COSMO's rotated pole grid, respectively.*

For the particular task of downscaling the 2m temperature (T2m), several predictor variables are chosen from the ERA5-dataset to encode the state of the planetary boundary layer in a data-driven way. This constitutes a notable difference to the Tier-1 dataset which only inputted the surface topography in addition to the coarse-grained T2m-field. By extending the set of predictor variables, the neural network is supposed to be applicable to arbitrary daytimes and seasons, whereas the neural network application with the Tier-1 dataset is limited to daytimes at noon of the summer half year. An overview of the predictor variables from the ERA5-dataset is provided in Table 8. Note that the usage of surface heat fluxes is only possible with the ERA5 short range forecasts that are initiated at 06 and 18 UTC, respectively, from the analysis fields within the 12-hour assimilation window (Hersbach et al., 2021). To reduce the footprint of spin-up effects, data between forecast hour six and 17 are used to construct the input data.

Similar to the Tier-1 dataset, the 2m temperature from the COSMO-REA6 dataset is complemented by the surface topography of the target domain. According to Sha et al., 2020, adding this static field to the predictands encourages the neural network to be orography-aware.

In total, the final dataset then comprises 111.548 samples of which data from 2017 and 2018 are used for the validation and testing (8748 samples each), respectively, while the remaining 11 years of data are reserved for training (94,052 samples). Note that this data split-approach maximises the statistical independence between the datasets in light of the autocorrelation of atmospheric data as argued in Schultz et al., 2021. The datasets are stored in netCDF-format and reside in about 45 GB of memory for the training data and 4.2 GB for the validation and the test dataset, respectively.

| Variable (variable name) | Data source (grid spacing) | Input/Output |
|---|---|---|
| **2m temperature (2t)** | ERA5 short-range fcst. ($\Delta x_{ERA5,rot} = 0.275°$) | input |
| **sensible surface heat flux (sshf)** | ERA5-short-range fcst. ($\Delta x_{ERA5,rot} = 0.275°$) | input |
| **latent surface heat flux (lshf)** | ERA5-short-range fcst. ($\Delta x_{ERA5,rot} = 0.275°$) | input |
| **boundary layer height (blh)** | ERA5-short-range fcst. ($\Delta x_{ERA5,rot} = 0.275°$) | input |
| **10m (u,v)-wind (10u, 10v)** | ERA5-short-range fcst. ($\Delta x_{ERA5,rot} = 0.275°$) | input |
| **850hPa and 925 hPa temperature (t_850, t_925)** | ERA5-short-range fcst. ($\Delta x_{ERA5,rot} = 0.275°$) | input |
| **surface geopotential (z)** | ERA5-short-range fcst. ($\Delta x_{ERA5,rot} = 0.275°$) | input |
| **2m temperature (t_2m)** | COSMO-REA6 ($\Delta x_{CREA6,rot} = 0.055°$) | target |
| **surface topography (hsurf)** | COSMO-REA6 ($\Delta x_{CREA6,rot} = 0.055°$) | target & input |

*Table 8: Overview of predictor (input) and predictand (target) variables used for the 2m temperature downscaling task of the Tier-2 dataset. In addition to the long names, the short names from the original grib-files are provided. A comprehensive overview of the original ERA5 is provided in ECMWF's documentation[7]. The COSMO-REA6 variable list can be obtained from DWD's OpenData-Server[8].*

### 3.5.2 ML solutions

For downscaling the coarse-grained ERA5-data to the high-resolved 2m temperature field of COSMO-REA6, we test the two neural network architectures which are described briefly in the following.

#### *3.5.2.1 U-Net*

A U-Net architecture is used as the baseline model for 2m temperature downscaling in application 5. The original U-Net was first proposed by Ronneberger et al., 2015, and used for biomedical image segmentation. This architecture deploys convolutional layers as building blocks and is capable of extracting features at various spatial scales using symmetrical encoder-decoder blocks. The U-Net architecture used in this study is adapted from Sha et al., 2020, and comprises three up- and down-sampling blocks for the encoder and decoder, respectively. Each down-sampling block consists of a convolutional layer followed by max-pooling to reduce the number of grid points in the horizontal plane. Skip connections connect the encoder with the decoder which applies deconvolutional layers to reverse the spatial data reduction of the encoder. A visualisation of the U-Net including some further information on the configuration is provided in Figure 10 (the U-Net constitutes the generator in WGAN). Note that this architecture has already been tested in scope of the Deliverable D1.1 and D2.3 where the Tier-1 dataset for a pure downscaling task was provided.

[7] https://confluence.ecmwf.int/display/CKB/ERA5%3A+data+documentation#heading-Parameterlistings
[8] https://opendata.dwd.de/climate_environment/REA/COSMO_REA6/

### *3.5.2.1 WGAN*

While the U-Net architecture turns out to perform reasonably well on the Tier-1 dataset in terms of the RMSE, the spatial variability of the downscaled 2m temperature field is underestimated. This is a well-known issue of pure convolutional network architectures since they are prone to produce blurry images in computer vision applications (see, e.g., Bulat et al., 2018).

To address this issue for the downscaling application on the Tier-2 dataset, we integrate the U-Net as a generator into a Generative Adversarial Network (GAN). In computer vision, GANs have become state-of-the art for super-resolution tasks since the study of Ledig et al., 2017. By incorporating a discriminator to distinguish between real and generated data in an adversarial optimization process, the generator is trained to fool the discriminator. By doing so, the generator is encouraged to produce (super-resolved) data that share the same statistical properties as real data which, in turn, implies that small-scale features have to be generated (Goodfellow, 2020).

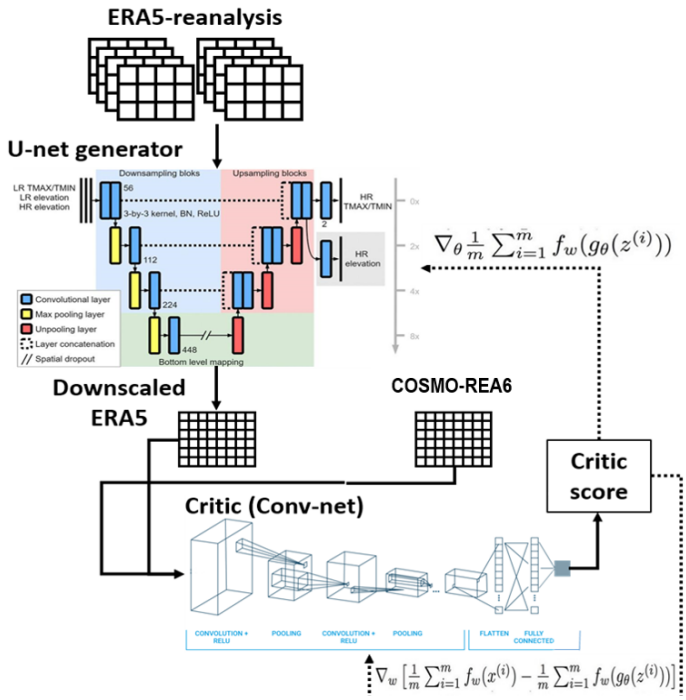


*Figure 10: The WGAN architecture deployed for the ERA5 to COSMO-REA6 downscaling task. The generator of the WGAN neural network constitutes the U-Net architecture proposed by Sha et al., 2020. The number of input channels for the down- and up-sampling blocks range from 56 to 448. The critic constitutes a convolutional network whose convolutional layers comprise 64 to 512 channels (four convolutional layers where the number of channels is doubled sequentially). The critic is optimised six times before the generator network gets trained in one iteration step. Note that the generator U-Net model is also tested as a standalone baseline model.*

Despite this appealing property, GANs are often hard to train since a balance between the generator and the discriminator must be retained throughout the training. To stabilise the optimization process, Wasserstein GANs (WGANs) have been proposed where the discriminator is replaced by a critic

(Arjovsky et al., 2017). Deploying a critic instead of a discriminator is beneficial since the critic does not saturate and thus is capable of providing clean gradients during training. By contrast, discriminators may suffer from vanishing gradients for which optimization gets stuck. While the aforementioned U-Net is deployed as the generator, a neural network comprising four convolutional layers, a global average pooling layer and a final dense layer is set-up as the critic for our downscaling task. The complete WGAN architecture is illustrated in Figure 10. In total, the WGAN model comprises about 5 million trainable parameters of which 1.5 million (3.5 million) are part of the critic (generator).

### 3.5.3 Experiments

In scope of this deliverable, three different experiments are conducted with the two presented neural network architectures: For Experiment 1 and 2, the Tier-2 dataset as described in Section 3.5.1 is fed to the U-Net and to the WGAN. The main focus of these experiments will be to check if the adversarial network architecture is capable of increasing the spatial variability of the downscaled 2m temperature field.

For Experiment 3, the training on WGAN is repeated with additional datetime embeddings for the input. The motivation for embedding the month of the year as well as the hour of the day is to ease encoding of the planetary boundary layer state in the neural network which is known to be a main driver of 2m temperature variability (see, e.g., Akylas et al., 2007).

For all experiments, the U-Net is trained for 30 epochs while the generator of WGAN is trained for 40 epochs. The initial learning rate $\lambda_{UNet}$ of the U-Net model is set to $1 * 10^{-4}$ which is then subject to exponential learning rate decay after five epochs (the decay exponent is set to -0.1). For the WGAN, different initial learning rates are set for the generator ($\lambda_{WGAN}^{gen} = 5 * 10^{-5}$) and the critic ($\lambda_{WGAN}^{critic} = 1 * 10^{-6}$). However, both learning rates are decayed between epoch five and ten after which $\lambda_{WGAN}^{gen} = 5 * 10^{-7}$ and $\lambda_{WGAN}^{critic} = 1 * 10^{-8}$ is applied to the backpropagation. The respective learning rates schedules have been obtained empirically from tuning.

The loss function of the U-Net constitutes the Mean Absolute Error, also known as L1-error, of the (normalised) 2m temperature. For the WGAN, the same reconstruction loss is weighted by a factor of 1000 and combined with the Wasserstein distance between the generated and real data. Furthermore, a gradient penalty (weighted by 10) is deployed for the WGAN to fulfil the 1-Lipschitz constraint.

### 3.5.4 Results

In the following, we briefly evaluate the experiments in terms of the RMSE, the bias and the spatial variability. For the latter, the domain-averaged amplitude of the horizontal gradient from the downscaled 2m temperature field is compared against the ground truth data. When the gradient

amplitude ratio equals 1, the local variability of both 2m temperature fields is the same, whereas it is underestimated (overestimated) for ratios smaller (larger) than 1.

#### 3.5.4.1 Experiment 1 and 2

The first row of Figure 11 shows the RMSE obtained with the U-Net (a) and with the WGAN (b) as a function of daytime for the testing data from 2018. The performance of both models is comparable, even though the U-Net attains marginally smaller RMSE-values ($RMSE_{UNet} = 1.11 \pm 0.1$ K vs. $RMSE_{WGAN} = 1.18 \pm 0.31\ K$). It is seen that the error tends to be larger at around local noon when the $RMSE$ approaches $1.5\ K$, whereas it is smallest overnight. Inspecting the bias shows that both neural networks attain a value close to 0 (not shown). Slightly larger systematic deviations are just present after midnight (cold bias) and in the morning hours (warm bias).

However, in terms of the spatial variability, the WGAN significantly outperforms the U-Net which underestimates the amplitude of the 2m temperature gradient by more than 10% on average (cf. Figure 11c,d). Especially between midnight and morning, the spatial variability is largely underestimated with the U-Net indicating that the 2m temperature field is too smooth. By contrast, WGAN nearly manages to recover the spatial variability of the ground truth data at early noon and only slightly underestimates it for other daytimes (except from a short period in the evening hours). Thus, the integration of the U-Net model as a generator to the WGAN turns out to be beneficial since the spatial variability gets substantially more realistic, while point-wise error metrics remain largely unaffected.

However, closer inspection of the results reveals persisting issues for the WGAN. Figure 12 (a) presents the diurnal cycle of the gradient amplitude ratio during summer, while Figure 12 (b) shows the temporally averaged RMSE over the target domain at 12 UTC for the same season. It is seen that the WGAN clearly underestimates the spatial variability over day and that the error is largest for the Alpine region. During this time, the PBL is typically well mixed due to strong solar heating which introduces a high spatial variability over complex terrain due to adiabatic or super-adiabatic near surface lapse rate. Obviously, the U-Net has problems to properly encode this PBL state from the input data, so that the spatial variability due to the underlying terrain is notoriously underestimated which also introduces systematic biases (not shown). In the following experiments, we therefore probe if additional daytime and season embedding (hour of the day and month of the year) can support proper encoding which in turn would improve the results for the summer season.

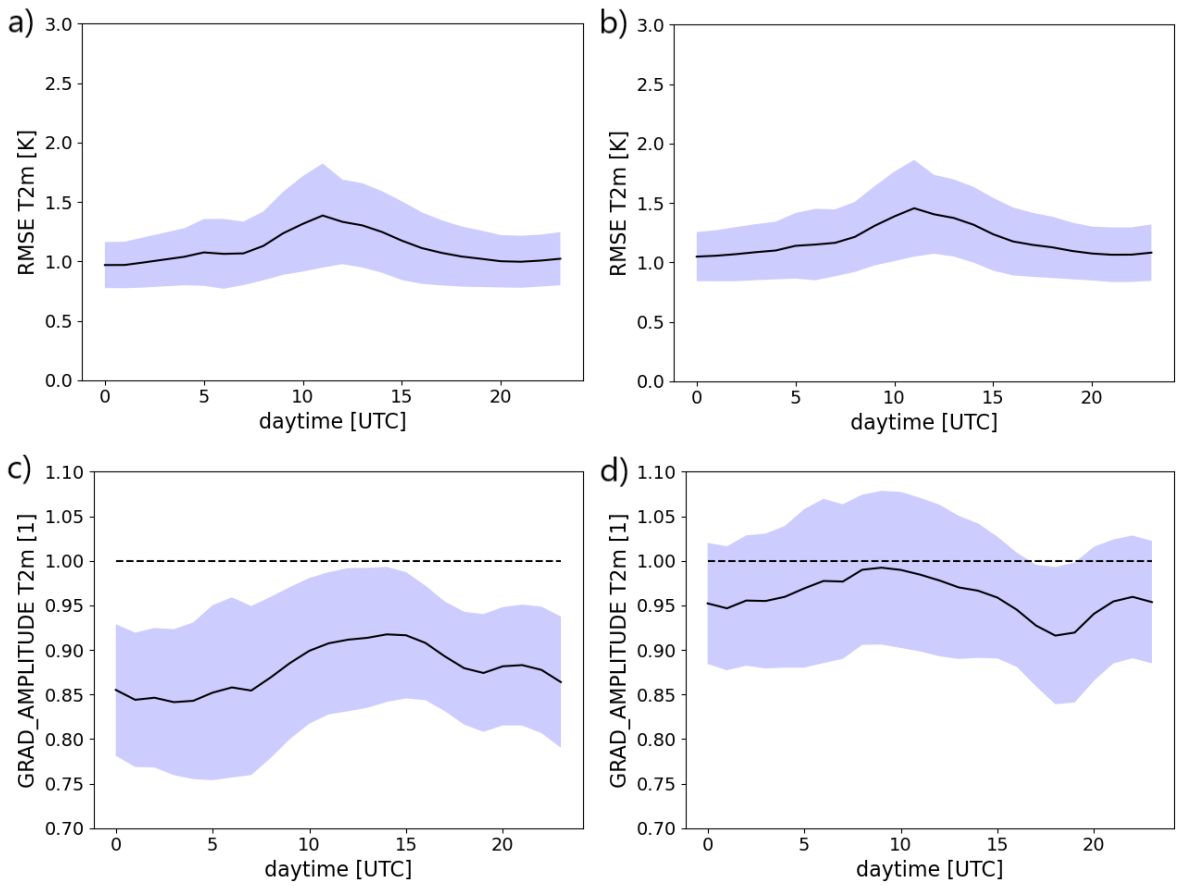


*Figure 11: The RMSE and the gradient amplitude ratio of T2m plotted against daytime of the trained downscaling neural networks for the test dataset are displayed in (a,b) and (c,d). The first column shows the results obtained with the U-Net, whereas the results of the WGAN are shown in the second row. The metrics are averaged over the complete target domain (see Figure 9) and the whole testing period (year 2018).*

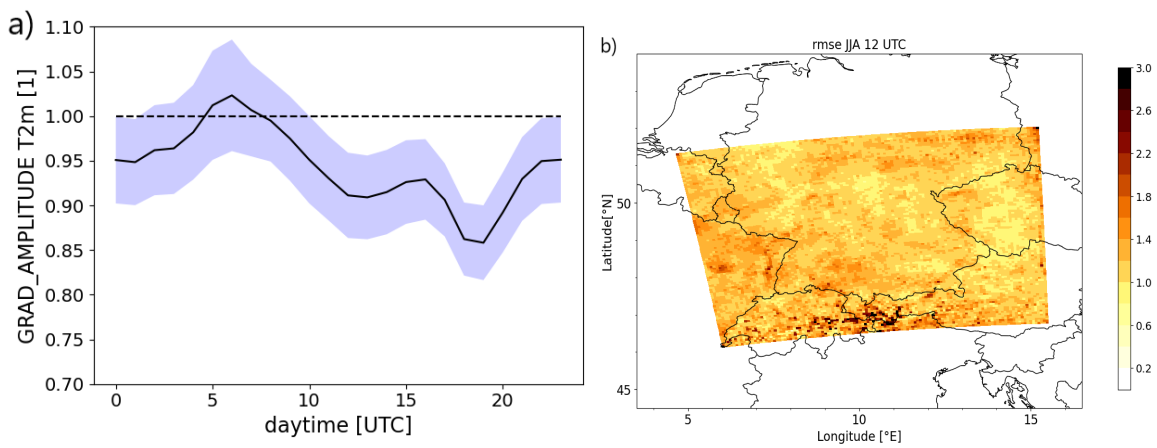


*Figure 12: (a) Averaged gradient amplitude ratio of T2m plotted against daytime for the summer months June, July and August (JJA). (b) Map plot of the averaged RMSE of T2m for the summer months JJA at 12 UTC*

### 3.5.4.2 Experiment 3

To embed the daytime and the season (date embedding), two additional input layers are added to the generator of the WGAN architecture. The hour of the day as well as the month of year corresponding to each sample are converted to one-hot vectors (of shape 24 and 12, respectively) which are then projected to a scalar value via a trainable, fully connected layer. By doing so, a data-driven encoding of the date-information can be learned by the neural network via backpropagation. Finally, the scalar value is expanded to a 2D-tensor to allow concatenation along the channel-dimension of the gridded ERA5-input data.

In general, the added value of the date embedding is fairly small. In terms of the RMSE and the gradient amplitude ratio of T2m, only small improvements can be diagnosed ($RMSE_{WGAN}^{embed} = 1.15 \pm 0.31$ K, gradient amplitude ratio $r_{WGAN}^{embed}(|\nabla \mathrm{T2m}|) = 0.970 \pm 0.083$). While the small improvements are consistently observed for most daytimes and seasons (cf. Figure 13), the above mentioned issues around noon during the summer months persist. The RMSE still tracks quite high during this time, although we note a more pronounced improvement of spatial variability (see Figure 14). All in all, further investigation is required to come up with a more robust added value of the datetime embedding.

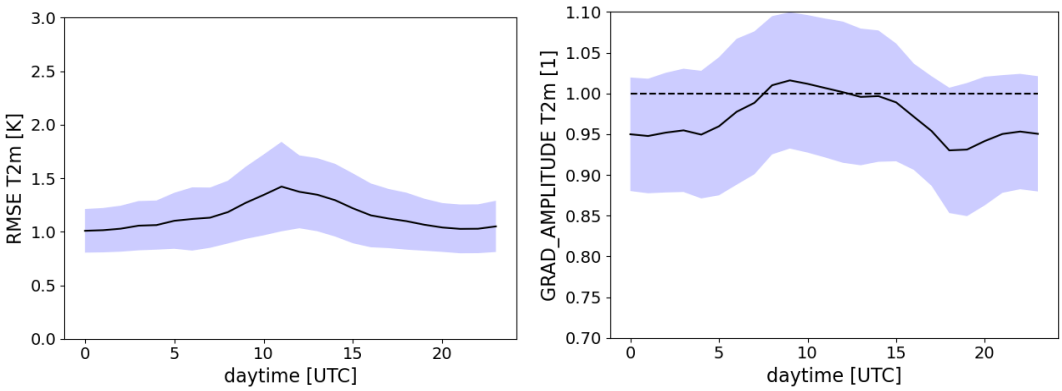


*Figure 13: (a) RMSE and (b) gradient amplitude ratio of T2m plotted against daytime of the WGAN with date embedding averaged over test dataset*

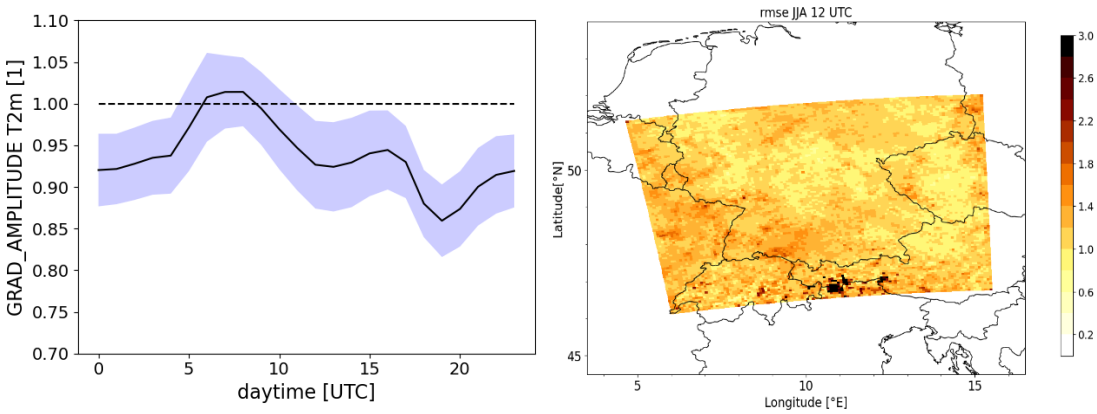


*Figure 14: As Figure 12, but for the trained WGAN with date embedding.*

### 3.5.5 Outlook

In the future, further effort to improve the performance of the downscaling models will be pursued. The main step comprises the following action items:

- The downscaling dataset will be completed by extending the training data back to 1995. Note that the current Tier-2 dataset is restricted to the period 2006-2018. The extension of the dataset will also require efficient data streaming from the netCDF-files. Currently, all the data are loaded into a memory which limits the application of the current source-code to HPC-systems equipped with large RAM. In the future, a more flexible input data pipe will be developed to relax the memory restrictions of the current approach.
- A systematic analysis on the added value of the predictor variables will be conducted. Additional input parameters such as the total cloud cover or predictor variables from previous time steps will be considered to improve the data-driven encoding of the PBL state which in turn is expected to improve the downscaling product.
- Further hyperparameter tuning for the deployed U-Net and WGAN will be undertaken. Other neural network architectures such as the downscaling network suggested by Leinonen et al., 2020, or Stengel et al., 2020, will also be considered as competing models.

In addition to improving the neural networks for downscaling the 2m temperature, we plan to extend the application of the ERA5 to COSMO-REA6 downscaling task to other meteorological fields. Specifically, downscaling of the incoming solar radiation or the wind at 10m above ground, which are both relevant for the renewable energy sector, will be tested as well. The overarching target is to provide a comprehensive benchmark dataset for downscaling that can be applied to different downscaling tasks. We note that an increased demand for such a benchmark dataset has recently arisen in the meteorological domain as, for instance, discussed in the *Machine Learning for Climate Science*-session of the EGU General Assembly 2022.

Finally, we also aim to extend our application to the challenging task of downscaling precipitation forecasts. In particular, we aim to make use of precipitation forecasts from the IFS HRES which constitutes the world-leading global NWP model operational at ECMWF. The IFS HRES forecasts are provided on a grid with $\Delta x_{IFS} \simeq 9$ km which is still too coarse to represent the high spatial variability of convective precipitation. Thus, rain-gauge adjusted radar observations provided by the RADKLIM dataset with $\Delta x_{RADKLIM} = 1$ km (Winterrath et al., 2017) will be used as the target of the downscaling task which will furthermore be extended to a probabilistic framework to account for the chaotic nature of precipitation processes. For this purpose, recent model architectures based on vision transformers such as the SwinR network will be tested and developed (Liang et al., 2021). One advantage of transformer-based models is that they can retain an optimised computational cost-accuracy trade-off. Our first results demonstrate that the training time of the SwinIR-downscaling model is much faster than the U-Net architecture. Preliminary results from a (deterministic) SwinIR-downscaling model together with the downscaled product from a U-Net model are provided in Figure 15. It is seen that the downscaled precipitation field is still too coarse for both models and that artefacts due to the shifting window-approach in SwinIR reside in the downscaled product (Zhang et

al., 2022). Thus, further development of the model architectures is required to improve the performance of the models, so that they can be compared to recent downscaling advances described in the study by Price and Rasp, 2022, and Harris et al., 2022.

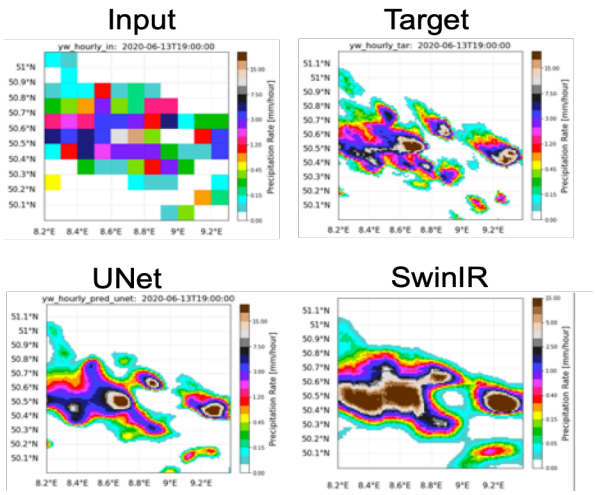


*Figure 15: Primary results of precipitation downscaling by using U-Net and SwinIR*

### 3.5.5 Data and Code access

The code can be accessed in our project repository[9] . The corresponding experiments can be run from the branch in Table 9. The code can be downloaded via a CliMetLab plugin

| Experiment # | Branch name |
|---|---|
| **1,2** | michael_issue#039_maelstrom_deliverable1.3 |
| **3** | bing_issue#41_embedding_date_and_time |
| **4, 5** | bing_issue#031_swinIR |

*Table 9: The branch names of the experiments for AP5*

## 3.6 AP6: Provide bespoke weather forecasts to support energy production in Europe

Forecasts for power production of renewable energy resources are prone to uncertainty. While forecasts for power production from solar resources usually exhibit low errors ($< 10\%$) that are quite stable in time, wind power forecasts typically suffer from larger uncertainties ($\gtrsim 10\%$) that have a higher variance.

To produce power production forecasts for individual plants and sites, the developed ML models consider small-scale, local weather conditions. The past has shown, though, that the forecast uncertainty may well be affected by large-scale weather patterns with time scales exceeding the maximum forecasting leadtime. As a consequence, we aim to investigate whether models informed

[9] https://gitlab.jsc.fz-juelich.de/esde/machine-learning/downscaling_maelstrom/-/tree/develop

by large-scale weather regimes (LSWRs) over Europe have the ability to reduce the uncertainty of power forecasting for wind and solar resources.

Here, we use an entirely data-driven approach to classify LSWRs. The data consist of a time series with multiple physical quantities that describe the weather conditions on multiple pressure levels on a large grid covering the whole of Europe. Hence, the dataset is high-dimensional. To apply a classification of weather patterns in the data, we perform a dimensionality reduction of the multi-level grid time series and apply a classification on the resulting low-dimensional projection of the data.

In the first approach, we want to study each individual LSWR found in the data and get information about their characteristics including stability and endurance. Here, we want to compare the different classes of LSWRs to the so-called Großwetterlagen (GWL) defined by the German Meteorological Service (Deutscher Wetterdienst, DWD).[10] The DWD created a catalogue of 40 GWL characterised by geopotential, temperature, relative humidity, and zonal and meridional wind component on multiple pressure levels. GWL has certain characteristics and are stable up to several days, and hence allow to generalise the evolution of small-scale weather conditions during the appearance of the respective GWL.

We then want to investigate how power production forecast uncertainty behaves for the different LSWRs since we suspect that there is a relation between LSWR classes and the prediction errors. Using this information, we aim to develop more advanced models that have knowledge about present LSWRs (LSWR-informed models), and thus, are more robust against changes in small-scale weather conditions.

### 3.6.1 Dataset

The underlying data are from the ECMWF IFS HRES model. A detailed description can be found in the document of the Deliverable D1.1 in Section 3.6.[11]

The data cover a time range of 2017-2020 with an hourly temporal resolution resampled to a daily resolution where we calculate the daily mean of each physical quantity. The grid covers the whole of Europe with an area of 35°N to 70°N and 25°W to 30°E and a spatial resolution of 0.1°N and 0.1°E. Each physical quantity is given on a set of multiple pressure levels (500hPa, 800hPa, 925hPa, 950hPa and 1000hPa).

Since we aim to compare the LSWRs that we find in the data to the GWL defined by DWD, we similarly use the following physical quantities at the 500 hPa pressure level:

- temperature $t$
- geopotential $z$
- relative humidity $r$
- zonal and meridional wind speed $u$ and $v$

[10] https://www.dwd.de/EN/ourservices/wetterlagenklassifikation/beschreibung.html
[11] https://www.maelstrom-eurohpc.eu/content/docs/uploads/doc6.pdf

### 3.6.2 Methodology

The classification of LSWRs is done by applying a clustering algorithm on the data that assigns each individual time step to a cluster, where outliers are allowed. Each cluster thus represents a unique LSWR that occured over the time span of the time series. Outliers here represent transition phases between different states of the system.

To allow classification of the high-dimensional time series grid data, we apply a dimensionality reduction algorithm (Principal Component Analysis, PCA). PCA finds a set of orthonormal vectors (principal components, PCs), where each of which accounts for a certain amount of variance of the dataset (if the data were projected onto that vector). Here, the fraction of the variance represented by each PC is in ascending order. I.e., the first component accounts for most of the variance in the dataset, the second for the second most, and so forth.

However, we only use a subset $N_{PCs}$ of the retrieved PCs to transform the original data into the low-dimensional PC space. The PC space is a multi-dimensional vector space that represents the phase space of the dynamical system described by the data. To avoid the curse of dimensions, we use only a reduced amount of PCs for the transformation such that the PCs reflect most of the variance of the data.

For clustering the states of the dynamical system in PC space, we use the *Hierarchical Density-Based Spatial Clustering Algorithm for Applications with Noise (HDBSCAN)* by Malzer and Baum (2019), which is a modification of the DBSCAN algorithm (Ester et al., 1996). As implied by the name, the algorithm finds high-density regions in a given data set that represent clusters of related data points. Data points outside of such high-density regions (outliers) are classified as noise.

#### *3.6.2.1 Dimensionality Reduction*

The PCA is performed on the whole dataset. To allow a PCA on a time series grid, the data have to be flattened by concatenating the rows (representing the different latitudes) of the grid for each physical quantity. The flattened grids of all investigated physical quantities are then concatenated as well.

However, we use only $N_{PCs} = 3$ to transform the data into PC space. This is because we only have $10^3$ data samples (4 years of data with roughly 365 days each). Choosing more PCs would require more data (*curse of dimensions*). I.e., using $N$ PCs would require at least $10^N$ data samples to allow a robust analysis since the clustering is performed on the $N$-dimensional phase space of the system.

#### *3.6.2.2 Clustering*

Before applying the clustering, the data are transformed into the 3-D subspace of the PC space. The result reflects the phase space containing all states of the dynamical system throughout the given time span (2017-2020).

Within this space, we perform a clustering to find recurring states where the system frequently resides in. Each retrieved cluster then represents a unique LSWR, i.e. all clusters represent the ensemble of

LSWRs that our system resided in during the given time range. Each data point in a cluster hence reflects the time steps within the time series where the LSWR of the respective cluster appeared. This allows a thorough statistical analysis of each LSWR.

#### *3.6.2.3 Statistical Analysis of the LSWRs*

The clusters (LSWRs) are then statistically analysed such that we retrieve information about the LSWRs in general (total abundance, mean and standard deviation of their duration). In addition, we plan to visualise the appearance of individual LSWR clusters at all time steps.

### 3.6.3 Results

For the dimensionality reduction via PCA we use $N_{PCs} = 3$. Figure 16 shows the scree plot for the first 5 PCs. The right axis (blue) shows the explained variance ratio of each PC. The left axis (red) shows the cumulative variance ratio reached for each PC. For our given dataset, the first three PCs account for $\sim 78\%$ of its variance. Thus, using 3 PCs already explains the majority of variance, although the dimensionality of the entire PC space is of order $10^3$. Hence, it is reasonable to use only the first 3 PCs for the dimensionality reduction of the data.

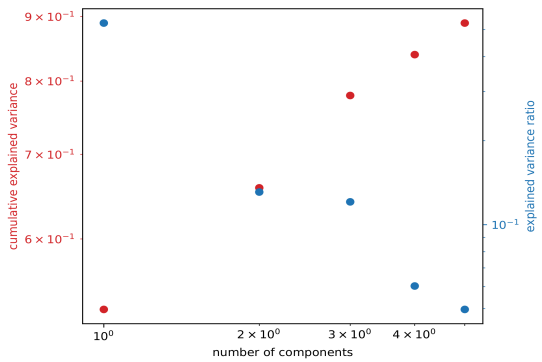


*Figure 16: Scree plot for the first 5 PCs*

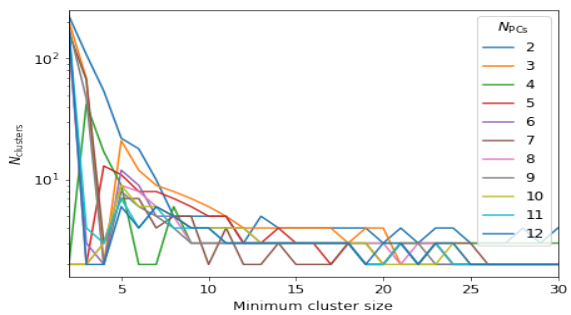


*Figure 17: Results of a hyperparameter study to investigate how the number of clusters found by HDBSCAN behaves for different amounts of PCs used for the dimensionality reduction and minimum cluster sizes as input for the HDBSCAN*

After transforming the data into the 3-dimensional PC space, applying the HDBSCAN yields $N_{clusters} = 3$ clusters, where $\sim 60\%$ of the data points (days) are classified as outliers (see Figure 17). Here, we use `min_cluster_size=10`. A hyperparameter study where we varied the number of PCs used for dimensionality reduction from 2–12 and the minimum cluster size from 2–30 has shown that the number of clusters found by the algorithm converges at a minimum cluster size of $\sim 10$ (see Figure 17).

The clusters represent areas in the low-dimensional phase space of our dynamical system where the system repeatedly resided in, i.e. the LSWRs we are actually interested in.

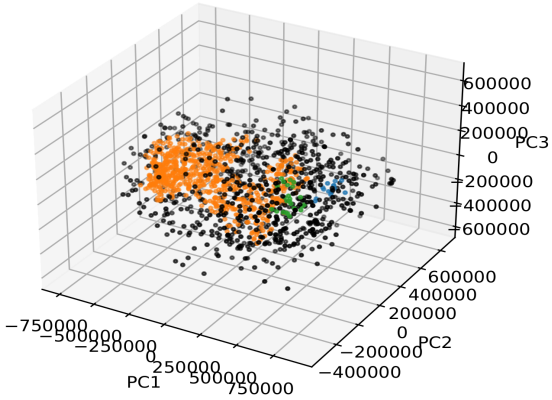


*Figure 18: Result of the HDBSCAN clustering on the data in the 3-dimensional PC space. The clusters are coloured in orange, green, and blue. Black data points are classified as outliers (noise)*

As shown in Figure 18, the algorithm finds one very large cluster (orange) and two smaller clusters (green and blue) that are not clearly separable by eyeball analysis, though. Hence, the procedure does not seem to yield robust results.

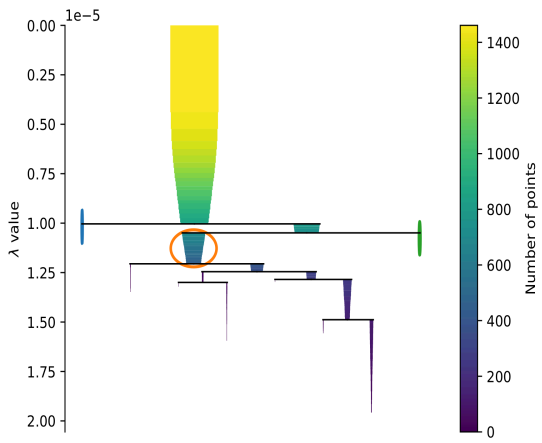


*Figure 19: Condensed tree plot of the HDBSCAN clustering. The three clusters found by the algorithm (orange, green, and blue) reflect those shown in Figure 3.6.3, colored respectively.*

Figure 19 reveals how HDBSCAN hierarchically splits the data points into smaller clusters. With decreasing core distance $d$ (or increasing $\lambda = 1/d$), the number of points identified as outliers (noise) increases, i.e. the size of the main cluster is reduced constantly since outskirt areas of the main cluster

get classified as clusters whose sizes are smaller than the minimum cluster size. The Figure 19 also shows that the maximum number of clusters that HDBSCAN is able to find is eight, where even a larger portion of the original dataset would be classified as noise than currently. Hence, even selecting the result of a DBSCAN with a small core distance would not give valuable results for our purpose.

### 3.6.4 Conclusion and Outlook

The result of the PCA and HDBSCAN is not very promising since we only find three clusters, which is not granular enough for our purpose. The three clusters may in fact represent larger weather circulation patterns. However, our aim is to find more clearly separable LSWRs with durations in the order of days. Moreover, the amount of data points identified as noise ($\sim 60\%$) is too large and drastically reduces the number of potential days that can be used for investigating any relationship of LSWRs to renewable power production.

Our next step will be to apply Dynamic Mode Decomposition (DMD) for dimensionality reduction and compare its results to that of the PCA. DMD is especially suited to find dominant modes in dynamical systems, originally developed for and applied to fluid mechanics by Schmid and Sesterhenn (2008). An advantage of DMD is that it is an entirely data-driven approach to analyse dynamical systems without requiring any information about the underlying equations of motion.

### 3.6.5 Data and Code Access

The data are available from the S3 bucket at ECMWF and can be downloaded using the CliMetLab `climetlab-maelstrom-power-production` plugin.[12,13] The code of the application is provided on the 4Cast GitHub account.[14] Instructions for usage are given there. The Jupyter notebook for the described methodology is located in the repository at `notebooks/pca.ipynb`.

[12] https://pypi.org/project/climetlab-maelstrom-power-production
[13] https://github.com/4castRenewables/climetlab-plugin-a6
[14] https://github.com/4castRenewables/maelstrom-a6

# 4 Distributed ML training on AP 3 and AP4

In this delivery, we also perform distributed deep learning on two of the applications: AP3 and AP4. In addition, we would like to evaluate the performance on different framework. Particularly, AP3 used the Horovod Framework, and performed distributed training of the neural network on A100 GPUs of the JUWELS Booster. AP4 conducted experiments on a single machine with four A100 GPUs with NVLink interconnects and used the PyTorch's built-in `DistributedDataParallel` module with NCCL as the underlying communication framework.

## 4.1 Results for parallel training of AP 3

We select the RNN architecture from experiment 2 to explore the possible benefits of distributed training. In the Table 10, we outline the results when training with 1, 2 & 4 GPUs, all located on a single node. The dataset is split into N (number of GPUs) subsets, and an epoch is defined as all processes passing through their subset of the data.

In preliminary experiments the initial learning rate was scaled linearly with the number of GPUs. However, for 4 GPUs this was found to be unstable in training, therefore the initial learning rate was halved to 0.002. Previous studies have shown that for large batches or distributed training a ramp-up phase for the early epoch learning was effective. In future work we will explore using this approach to maximise training efficiency.

Training with multiple GPUs decreases time to solution, although we find a noticeably sublinear relationship between GPU number and walltime. The scores on the training set are slightly degraded when training uses multiple GPUs.

Future work will explore methods to improve final test scores.

| GPUs | Initial learning rate | Epoch | Batch/ GPU | Loss | SW MSE | SW MAE | SW HR MSE | SW HR MAE | Training wall-time (s) |
|---|---|---|---|---|---|---|---|---|---|
| **1** | 0.001 | 50 | 512 | 0.001718 | 1.618 | 0.4290 | $3.959*10^{-4}$ | 0.006129 | 65189 |
| **2** | 0.002 | 50 | 512 | 0.001991 | 1.806 | 0.4655 | $6.319*10^{-4}$ | 0.007597 | 46776 |
| **4** | 0.002 | 50 | 512 | 0.002061 | 1.950 | 0.4889 | $6.258*10^{-4}$ | 0.007802 | 22724 |

*Table 10: Computational performance of distributed training for experiment 2 of AP3*

## 4.2 Results of parallel training on AP 4

We evaluate training of AP 4 using the U-Net model with ten ensemble members and evaluating CRPS for 2-meter temperature (T2m). See Section 3.4 for additional details. All experiments used the same initial learning rate, and experiments with more than one GPU followed the linear learning rate scaling rule with a warmup of five epochs to preserve the accuracy as suggested by Goyal et al., 2017. Results are reported for a single experiment for each configuration.

Accuracy appears to degrade slightly with additional GPUs (and, hence, a larger batch size), while training time improves significantly, albeit with less than perfect scaling. In future work we expect to better tune hyperparameters to mitigate accuracy degradation.

| #GPUs | Initial learning rate | Target learning rate | Samples/GPU | T2m CRPS | Time [s] |
|---|---|---|---|---|---|
| **1** | $10^{-5}$ | $10^{-5}$ | 8 | 0.640 | 3501 |
| **2** | $10^{-5}$ | $2*10^{-5}$ | 8 | 0.644 | 2421 |
| **4** | $10^{-5}$ | $4*10^{-5}$ | 8 | 0.651 | 1581 |

*Table 11: Distributed training results across 1, 2 and 4 GPUs for AP4*

# 5 ML framework comparison: TensorFlow 2 VS PyTorch on App 5

In this section, we evaluate the performance of two ML frameworks: TensorFlow 2 and PyTorch. We used the Tier 2 dataset of AP5, and trained U-Net architecture as described in Section 3.4. The architecture has been implemented for both ML frameworks. We keep the hyperparameters (e.g., batch size, learning rate, epochs etc.) the same for training the neural network, and run three times for each both frameworks to capture the variability of the performance.

Table 13 demonstrates the average times for data generator creation, the first training epoch of the three runs, the average time of 30 epochs, and the total training time. We find that the overall performance for PyTorch is better than for TensorFlow. The total training time for PyTorch is around 2 minutes shorter when compared to TensorFlow. Notably, the time spent on the 1$^{st}$ training epoch is significantly shorter for PyTorch. This means the performance for the initialisation of the model for PyTorch is better. Remember that we are only using a simple neural network - U-Net – for these tests. The benefit of using PyTorch might be gained when the advanced architecture with more parameters is used.

| Time [s] | PyTorch | TensorFlow |
|---|---|---|
| **Creating data generator** | 0.00 | 0.03 |
| **1st training epoch time** | 110 | 185 |
| **Average time / training epoch.** | 93 | 98 |
| **Total training time** | 2794 | 2926 |

*Table 12: Computational performance of ML frameworks, TensorFlow and PyTorch, on AP5*

# 6 Benchmarking with various numerical precision on App 3

Reduced precision was explored using the TensorFlow option to utilise TensorFloat32. TensorFloat32 format is available only on NVIDIA GPUs (here the A100 was used). This format combines the significand (precision) of a half-precision number (10 bits) with the mantissa (dynamic range) of a single precision number (8 bits). For certain applications this format was shown to give a 2x speed-up versus single precision. Here we explore how this numerical format can aid training performance for our benchmark.

Control of TensorFloat32 can be easily achieved. In TensorFlow version 2.6 the control is set with

```
tf.config.experimental.enable_tensor_float_32_execution
```

We explored the impact on all 3 experiments outlined in section 3.3, which can be seen in the Table 14. Due to computational cost, each entry is based on a single simulation, so small fluctuations in the test loss should be discounted.

| Architectures | Average epoch time (s) without TF32 | Average epoch time (s) with TF32 | Test loss without TF32 | Test loss with TF32 |
|---|---|---|---|---|
| **CNN** | 920 | 710 | 0.027 | 0.030 |
| **RNN** | 1310 | 1245 | 0.0016 | 0.0017 |
| **CNN+attention** | 1994 | 1468 | 0.0014 | 0.0040 |

*Table 13: Experiments results on testing different precision*

Experiment 1 shows a nontrivial improvement in epoch time (~22% decrease) with only a small change in test loss (likely not statistically significant). However, experiment 1 was the weakest learner of our benchmarks. Experiment 2 shows a marginal improvement in performance when using TF32, with minimal degradation in the test loss scores. Experiment 3 shows a more significant increase in performance (~25%), but a significantly degraded final test loss. This will require further exploration to understand and improve performance with self-attention layers at reduced numerical precision.

In future work we will explore more reduced precision options (e.g. use of 16-bit precision). This will be explored across more of the applications.

# 7 References

Akylas E, Kotroni V, Lagouvardos K. Sensitivity of high-resolution operational weather forecasts to the choice of the planetary boundary layer scheme. Atmospheric Research. 2007 Mar 1;84(1):49-57.

Ashkboos, Saleh, Huang, Langwen, Dryden, Nikoli, Ben-Nun, Tal, Dueben, Peter, Gianinazzi, Lukas, Kummer, Luca, and Hoefler, Torsten. "ENS-10: A Dataset For Post-Processing Ensemble Weather Forecast." arXiv preprint arXiv:2206.14786 (2022).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. “Attention is all you need.” NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017 (pp. 6000–6010)

Bollmeyer C, Keller JD, Ohlwein C, Wahl S, Crewell S, Friederichs P, Hense A, Keune J, Kneifel S, Pscheidt I, Redl S. Towards a high-resolution regional reanalysis for the European CORDEX domain. Quarterly Journal of the Royal Meteorological Society. 2015 Jan;141(686):1-5.

Bollmeyer C. A high-resolution regional reanalysis for Europe and Germany (Doctoral dissertation, Universitäts-und Landesbibliothek Bonn).

Chen LC, Papandreou G, Kokkinos I, Murphy K, Yuille AL. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence. 2017 Apr 27;40(4):834-48.

Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, Uszkoreit J. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929. 2020 Oct 22.

Gneiting, Tilmann, Raferty, Adrian E., Westveld III, Anton H., and Goldman, Tom. “Calibrated Probabilistic Forecasting Using Ensemble Model Output Statistics and Minimum CRPS Estimation.” Monthly Weather Review, 133(5):1098-1118, 2005.

Goyal P, Dollár P, Girshick R, Noordhuis P, Wesolowski L, Kyrola A, Tulloch A, Jia Y, He K. Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677. 2017 Jun 8.

Giorgi F, Gutowski Jr WJ. Regional dynamical downscaling and the CORDEX initiative. Annual review of environment and resources. 2015 Nov 4;40:467-90.

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial networks. Communications of the ACM. 2020 Oct 22;63(11):139-44.

Grönquist, Peter, Yao, Chengyuan, Ben-Nun, Tal, Dryden, Nikoli, Dueben, Peter, Li, Shigang, and Hoefler, Torsten. “Deep Learning for Post-Processing Ensemble Weather Forecasts.” Philosophical Transactions of the Royal Society A, 379(2194):20200092, 2021.

Hersbach H, Bell B, Berrisford P, Hirahara S, Horányi A, Muñoz-Sabater J, Nicolas J, Peubey C, Radu R, Schepers D, Simmons A. The ERA5 global reanalysis. Quarterly Journal of the Royal Meteorological Society. 2020 Jul;146(730):1999-2049.

Harris L, McRae AT, Chantry M, Dueben PD, Palmer TN. A Generative Deep Learning Approach to Stochastic Downscaling of Precipitation Forecasts. arXiv preprint arXiv:2204.02028. 2022 Apr 5.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1. 2019 (pp. 4171–4186)

Leinonen J, Nerini D, Berne A. Stochastic super-resolution for downscaling time-evolving atmospheric fields with a generative adversarial network. IEEE Transactions on Geoscience and Remote Sensing. 2020 Nov 2;59(9):7211-23.

Liang J, Cao J, Sun G, Zhang K, Van Gool L, Timofte R. Swinir: Image restoration using swin transformer. InProceedings of the IEEE/CVF International Conference on Computer Vision 2021 (pp. 1833-1844)

Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, Lin S, Guo B. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision 2021 (pp. 10012-10022).

Malzer, C. and Baum, M., “A Hybrid Approach To Hierarchical Density-based Cluster Selection”. arXiv e-prints, 2019.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise”. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96). AAAI Press, 226–231, 1996.

Price I, Rasp S. Increasing the accuracy and resolution of precipitation forecasts using deep generative models. InInternational Conference on Artificial Intelligence and Statistics 2022 May 3 (pp. 10555-10571). PMLR.

Pengcheng He, Xiaodong Liu, Jianfeng Gao and Weizhu Chen. “DEBERTA: DECODING-ENHANCED BERT WITH DISENTANGLED ATTENTION.” International Conference on Learning Representations. 2021a.

Pengcheng He, Jianfeng Gao and Weizhu Chen. “DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing.” arXiv preprint:2111.09543. 2021b.

Ramachandran, Prajit, Barret Zoph, and Quoc V. Le. "Searching for activation functions." arXiv preprint arXiv:1710.05941 (2017).

Ronneberger, Olaf, Fischer, Philipp, and Brox, Thomas. “U-Net: Convolutional Networks for Biomedical Image Segmentation.” International Conference on Medical Image Computing and Computer-Assisted Intervention, pages 234–241. Springer, 2015.

Sha Y, Gagne II DJ, West G, Stull R. Deep-learning-based gridded downscaling of surface meteorological variables in complex terrain. Part I: Daily maximum and minimum 2-m temperature. Journal of Applied Meteorology and Climatology. 2020 Dec;59(12):2057-73.

Stengel K, Glaws A, Hettinger D, King RN. Adversarial super-resolution of climatological wind and solar data. Proceedings of the National Academy of Sciences. 2020 Jul 21;117(29):16805-15.

Schmid, P. & Sesterhenn, J., “Dynamic Mode Decomposition of numerical and experimental data”, Journal of Fluid Mechanics. 656. 10.1017/S0022112010001217, 2008.

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. Advances in neural information processing systems. 2017;30.

Winterrath, Tanja; Brendel, Christoph, Hafer, Mario; Junghänel, Thomas; Klameth, Anna; Lengfeld, Katharina; Walawender, Ewelina; Weigl, Elmar; Becker, Andreas (2018): RADKLIM Version 2017.002: Reprocessed gauge-adjusted radar data, one-hour precipitation sums (RW) DOI: 10.5676/DWD/RADKLIM_RW_V2017.002

Zhang B, Gu S, Zhang B, Bao J, Chen D, Wen F, Wang Y, Guo B. Styleswin: Transformer-based gan for high-resolution image generation. InProceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2022 (pp. 11304-11314).

Zamo, Michaël, and Philippe Naveau. "Estimation of the continuous ranked probability score with limited information and applications to ensemble weather forecasts." Mathematical Geosciences 50.2 (2018): 209-234.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, Quoc V. Le. “XLNet: generalized autoregressive pretraining for language understanding.” NIPS'19: Proceedings of the 33rd International Conference on Neural Information Processing Systems. 2019 (pp. 5753–5763)

## Document History

| Version | Author(s) | Date | Changes |
|---|---|---|---|
| **0.1** | Bing, Gong (FZJ) | 16/09/2022 | Initial version |
| | | | |
| | | | |
| | | | |
| | | | |

## Internal Review History

| Internal Reviewers | Date | Comments |
|---|---|---|
| **Peter Dueben (ECMWF)** | 18/09/2022 | Returned with minor comments and suggestions |
| **Mats Brorsson (University of Luxembourgh)** | 21/09/2022 | Returned with minor comments and suggestions |
| **Frederica Gualdi (E4)** | 23/09/2022 | Returned with minor comments and suggestions |

## Estimated Effort Contribution per Partner

| Partner | Effort |
|---|---|
| **FZJ** | 4 PM |
| **MetNor** | |
| **4cast** | |
| **ECMWF** | 1 PM |
| **ETH** | |
| **Total** | **5** |