MAchinE Learning for Scalable meTeoROlogy and climate

Revisions of customised ML solutions for enhanced datasets

www.maelstrom-eurohpc.eu

MAELSTROM

D1.4: Revisions of customised ML solutions for enhanced datasets

Author(s):	Michael Langguth (FZJ), Bing Gong (FZJ), Martin Schultz (FZJ) and all application developers of Work Package 1

Dissemination Level:	Public
Date:	15/09/2023
Version:	0.1
Contractual Delivery Date:	30/09/2023
Work Package/ Task:	WP1/ T1.4
Document Owner:	FZJ
Contributors:	4cast, ECMWF, ETH, FZJ, MetNor
Status:	Final

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

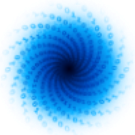
Research and Innovation Action (RIA)
H2020-JTI-EuroHPC-2019-1: Towards Extreme Scale Technologies and Applications

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

Published by the MAELSTROM Consortium

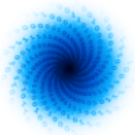
Contact:
ECMWF, Shinfield Park, Reading, RG2 9AX, United Kingdom
Peter.Dueben@ecmwf.int

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

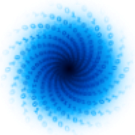


Contents

Figures

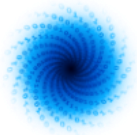


Tables



1 Executive Summary

This deliverable provides the description of the updated machine learning solutions for the MAELSTROM applications that have been developed based on the enhanced benchmark datasets described in Deliverable 1.2 and 1.3 (D1.2 and D1.3). A general overview on the progress and a discussion of deviations from the proposal is provided in Section 2. There has been no change of plans for the application development and, overall, the work on the MAELSTROM applications is developing nicely (see detailed presentation in Section 3). However, MAELSTROM has supported two more application areas – the development of full machine learned weather forecast models and the development of Foundation Models for weather and climate applications – as both areas have shown very fast developments and significant impact in the last two years and as they are very interesting for MAELSTROM's co-design approach as the developed tools are very data and compute intensive. Finally, Section 4 is presenting an example of how the Mantik tool, developed in work package 2, can be used for the developments of machine learning applications. Application 2 (AP2) from the set of MAELSTROM applications serves as an example here.



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.

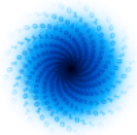
The MAELSTROM software framework will enable scientists to apply and compare machine learning tools and libraries efficiently across a wide range of computer systems. A user interface will link application developers with compute system designers, and automated benchmarking and error detection of machine 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 and work performed in this deliverable

To close the MAELSTROM co-design cycle between application, software and hardware developments and Work Package 1, 2 and 3 for the second time, this deliverable provides an update on the developments of the MAELSTROM applications. The progress that has been achieved is documented per MAELSTROM application in Section 3. The developments are based on the hardware benchmarks that were performed in Work Package 3 (see D3.4 and D3.6) and some of the applications are also already making active use of the software tools of Work Package 2 (see D2.3 and D2.4) as documented for AP2 in Section 4.



2.2.2 Deviations and counter measures

Two developments have recently changed how machine learning will impact the weather and climate domain in the medium and longer term future that were not foreseeable during the writing phase of the MAELSTROM proposal.

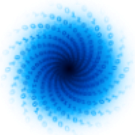
The first development is the great success of entire weather prediction systems based on machine learning that are (yet) trained from the ERA5 reanalysis dataset. The models are now showing a level of quality that is beating scores for deterministic weather predictions of conventional models, therefore challenging the entire way of how numerical weather prediction is formed. Big technology companies such as Google/Deepmind, NVIDIA, Huawei and Microsoft are building their own model configurations. The MAELSTROM applications are in general following a slightly different motivation when compared to the full machine learning weather forecast models as they advance conventional models with machine learning in a hybrid machine learning / conventional approach, for example via post-processing or via the emulation of model components. The new machine-learned weather models do not make the MAELSTROM applications obsolete as hybrid approaches for numerical weather prediction will still play a very important role for the future of numerical weather predictions. Therefore, the development of the six MAELSTROM applications have proceeded and the results are documented below. However, MAELSTROM has embraced the new developments supporting the evaluation and comparability of machine learned weather forecast models. The level of support is consistent with MAELSTROM's ideology to make machine-learned approaches more comparable. Two publications have been supported by MAELSTROM in this context.

Ben-Bouallegue et al. 2023 is performing a detailed evaluation to understand how the Pangu Weather machine learning model that was developed in *Bi et al. 2023* compares to results from conventional weather models. This is important as the new models use very different concepts for numerical discretisation and time stepping techniques, and as weather predictions need to be trustable to be useful with a good understanding of limits in predictability that cannot easily be derived from deterministic forecast scores.

Rasp et al. 2023 is publishing a new benchmarking framework for pure machine learning models that provides access to training data and a detailed comparison of diagnostics to provide a fair comparison of results for different machine-learned weather forecast models. The paper provides a new baseline for model comparisons and follows the concept of MAELSTROM as large datasets are made available to the public for download with easy access facilitated by notebooks.

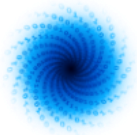
To engage with the pure machine learning model developments was also important for MAELSTROM as the developed models are among the largest machine learning tools that are currently developed in the weather and climate domains. Both in terms of data but even more importantly in terms of parallel training.

The second development was the move towards machine learning foundation models in many domains. These models are very large machine learning applications that follow a different philosophy when compared to machine-learned tools trained for a specific application, as they use semi-supervised training to learn the basic features and connectivity of the data in a pretraining step. In a second step, the learned abstraction of the data can be used for fine-tuning for specific tasks and applications. This approach leads to tools that are more generic compared to tools from direct



supervised training. Furthermore, foundation models often provide better results when compared to specific tools for applications in many domains, and in particular for large language models.

The domain of weather and climate is starting to test whether foundation models can be useful for weather and climate applications and, if successful, foundation models may lead to yet another step-change in the quality of machine learning tools. The approach is of particular interest for MAELSTROM as the resulting models need very large scale training data sets and large-scale machine-learned models that need significant high performance compute power to be trained. MAELSTROM has therefore supported one of the first approaches to build a foundation model for weather and climate science, called AtmoRep (Lessig et al. 2023), and AtmoRep is now also tested for use in Application 5 (see Section 3.5).



3 Update on Advanced Machine Learning solutions for the six applications

3.1 AP1: Blend citizen observations and numerical weather forecasts

The goal of AP1 is to produce high resolution (1x1 km) hourly temperature forecasts with a lead time of up to 58 hours for the Nordic countries, using NWP forecasts as inputs and recent conditions as estimated by citizen observations. In the context of the recent developments in purely data-driven forecasting, this application is a hybrid approach, leveraging both conventional post-processing of NWP output and including information from a starting state based on available observations that the ML-model can propagate forward in time.

The aim of this deliverable is to fine-tune the configuration of the U-Net (Fig. 1) in order to improve its predictive accuracy and make it ready for operational use at MET-Norway. The U-Net model was the most promising of the architectures studied in Deliverable 1.3 (D1.3). In the previous deliverable on scientific testing (D1.3), we noted that the data loading pipeline needed to be improved to allow for extensive testing of the model. A major outcome of the co-design loop with WP2 and WP3 was a much improved data processing pipeline. The work, culminating in D3.6, resulted in an improvement in the average processing performance from 0.42 GB/s (in D1.3) to 3.2 GB/s. This was a result of better exploitation of hardware resources, including CPUs, GPUs, and memory. More details on these improvements are documented in D2.4. We use these gains to greatly reduce the cost of training a suitable model.

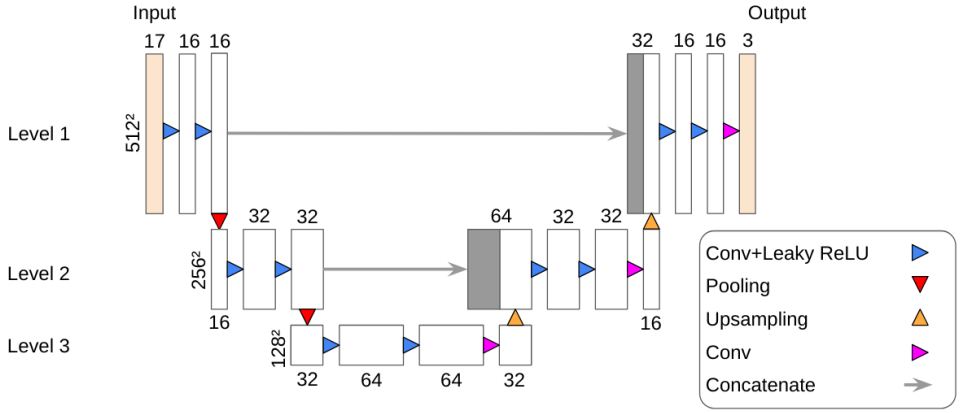
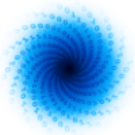


Fig. 1: Schematic diagram of the U-Net used in this application.

3.1.1 Dataset

The dataset has not changed significantly since D1.3. The dataset is still 6TB and contains the same predictors (14). The main change is that lead time-independent predictors are placed in a separate variable that does not contain the lead time dimension. This was done to make the dataset more intuitive for others to use.



The full dataset has two years of daily files. We use the first year (2020-03 to 2021-02) as the training dataset, where 24 files have been reserved for validation. We use the remaining time period (2021-03 to 2022-02) for final testing.

As the U-Net can introduce artefacts near the borders of the domain it is applied on, we removed a band of 32 pixels in width around the border when computing the loss function. This was done in training, validation, and testing.

3.1.2 Iterating towards a working U-Net model

We spent a significant effort getting a basic configuration of the U-Net model to work properly with our data. Early attempts appeared to give good validation scores with realistically looking output fields. However, we determined that this was not a result of the network's ability to capture signals at different spatial scales. All value was provided by the skip-connections on the top level and the lower levels did not provide any fields that impacted the final output fields. Our complex U-Net was effectively just a simple convolutional neural network. We checked this by plotting examples of tensors as it passed through the different levels in the network and noted that the final upsample fields (between level 2 and level1) were all 0.

We believe this problem is caused by the ReLU activation layers, which sets negative inputs to 0 and keeps positive inputs unchanged. Under certain conditions, these units can go dormant failing to provide a gradient that helps the network learn new features. This is referred to as the "dying ReLU" problem. We noticed that as the training progressed, more and more ReLU units died. We also tested exponential linear units, but this gave similar problems, possibly due to vanishing gradients when input values are very negative. We found that leaky ReLU activation layers solve this problem. With this type of activation, we got coarse grained fields as shown in Fig. 2.

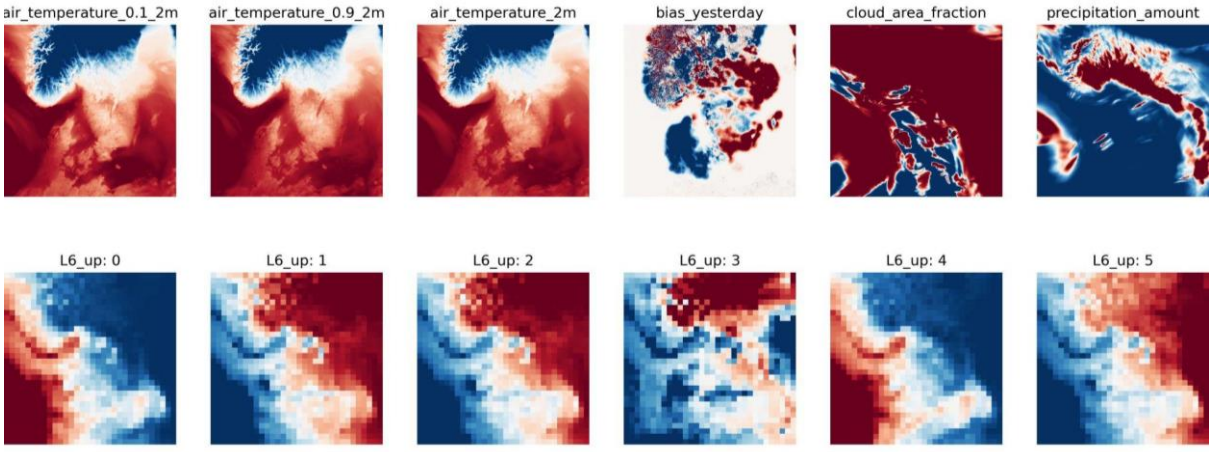
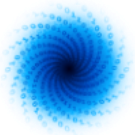


Fig. 2: Example input fields (upper row) and fields provided by the upsampling layer in Level 6 (lower row) for the U-Net model for a 512x512 patch covering Denmark and southern Norway and Sweden. The first 6 fields for each type are shown.

Leaky ReLU caused exploding gradients. This was solved by limiting the L2 norm of the gradients to 1, which is easily implemented by providing an argument to the Adam optimizer in Keras. This does,



however, come with a sizable performance hit (on the order of 20%). We did not investigate how to mitigate this performance hit.

We also investigated batch normalisation, which is commonly used in U-Nets after each convolution layer, but before the activation layer. This did not lead to stable results for us. We used BatchNormalization in Keras, which computes running mean and variance of the inputs to the layer. We tested different values of momentum (which governs how fast values are adapted). We suspect that part of the problem is that the batches we provide are not large enough and do not contain samples that are sufficiently random. Our data loader is not designed to give a completely random sample in each batch. It is possible that due to this, there is a coordinate shift throughout the epoch and that the batch normalisation keeps lagging behind.

In addition, we tested different methods for splitting the dataset into chunks that are used by each Horovod process. Each process computes the gradient of a given batch of data, and the gradients are averaged. By splitting the dataset such that each process gets 3 consecutive months of data, we ensure that all seasons are represented each time a batch is processed. We believe this should lead to faster learning.

After a lot of trial and error, we came up with a basic U-Net model with the following configuration:

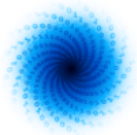
- 6 levels
- 1x1 convolution size
- Splitting the whole domain into patches of size 512x512
- Batch size 1
- 16 features on level 1
- Feature factor of 2 (the factor increase in number of features as you increase the level)
- 2x2 max pooling
- Leaky ReLU activation after each convolution
- 2x2 nearest neighbour upsampling
- Cosine decay restarts learning rate schedule, initial learning rate 1.0e-3
- Include extra static predictors: forecast lead time, x/y spatial coordinates

This configuration has 1,314,019 trainable parameters.

3.1.3 Hyper-parameter tuning

We tested different settings of the basic U-Net configuration. The hyper-parameter space is quite large and we cannot test every combination. We therefore perturb some of the hyper-parameters of this model, one at a time to see the effect of each parameter (Table 1).

Hyper-parameter	Basic configuration	Alternative configuration
Pooling operator	Max	Average



Number of levels Pooling size	6 2x2	3 4x4
Feature ratio	2	1
Convolution size	1x1	3x3

Table 1: List of hyper-parameters that were tested.

We used the Juelich Benchmarking Environment (JUBE) to organise the hyper-parameter tuning. This is similar to what we did in D3.6 with testing different hardware and data processing settings. We scheduled the jobs on a single JUWELS Booster node, performing distributed training on 4 Nvidia A100 GPUs.

Each training run traverses the training dataset three times. We computed the validation score f 30 times throughout the training period and stored the best performing weights. This model state was then used for final testing.

One important aspect of AP1 is its high resolution grid (1x1km). This means we need many levels in the U-Net to ensure a broad receptive field. However, this leads to many parameters. For example, a 6 level U-Net has over 1 million parameters. We therefore tested a U-Net model where the downsampling ratio is 4 instead of 2. Then, we can get the same receptive field with 3 levels as with 6 levels. We also changed the 3x3 convolution to a 1x1 convolution.

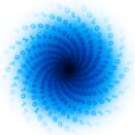
Because the basis for the input data source (2.5km) differs from the final output grid (1km), there is an elevation difference that can trivially be corrected for. We created a benchmark model that copied the raw forecasts and allowed for an elevation correction with a trainable constant lapse rate. This was trained to be 7.39°C/km.

We ran each configuration three times and averaged their test loss.

Description	Parameters	Test loss	Improvement
Raw elevation corrected model	1	0.2495	0.0 %
Basic configuration	1,314,019	0.2170	13.0%
Fewer features (feature ratio 1)	13,891	0.2170	13.0%
Mean pooling instead of max pooling	1,314,019	0.2139	14.3%
Fewer levels (3), bigger downsampling ratio (4)	20,195	0.2174	12.9%
Bigger convolution stencil (3x3)	9,000,291	0.2193	12.1%

Table 2: The improvement column shows the fractional decrease in test loss relative to the raw elevation corrected model.

These results indicate that all models performed quite well, leading to improvements over the raw elevation-corrected forecast. The main tuning benefit was to replace the max pooling layer with a mean pooling layer, leading to a further 1.3% improvement. Max pooling is often used to find well-



defined edges in images and it is possible that in this application, large scale averages provide more relevant signals. The other tuning efforts did not lead to any improvement over the basic configuration.

3.1.4 Assessing the impact of predictor variables

The input data set contains a wealth of predictors. For operational implementation, including extra variables causes several challenges. Firstly, they can slow down production because the features can be expensive to extract from the input model data. Secondly, they make the model more vulnerable to changes in the NWP model. The physics routines in the NWP model we use are tuned over time. Including a greater number of variables increases the risk of inconsistency between the training dataset and the data used in operations. We therefore want to make sure that the input variables do provide added value.

To do this, we trained the final U-Net model on different subsets of the parameters. In one experiment, we used only the temperature variables (and static predictors such as altitude, lead time, and x/y coordinates). In the other configuration, we added on the two bias variables (recent bias and yesterday's bias). Results indicate that performance is greatly reduced when the two bias variables were omitted (Fig. 3). We also note a reduction in performance when the non-temperature and bias related variables were omitted (difference between black and blue line). Thus, winds, precipitation, and clouds also provide added value to our model, though to a lesser extent than the bias predictors.

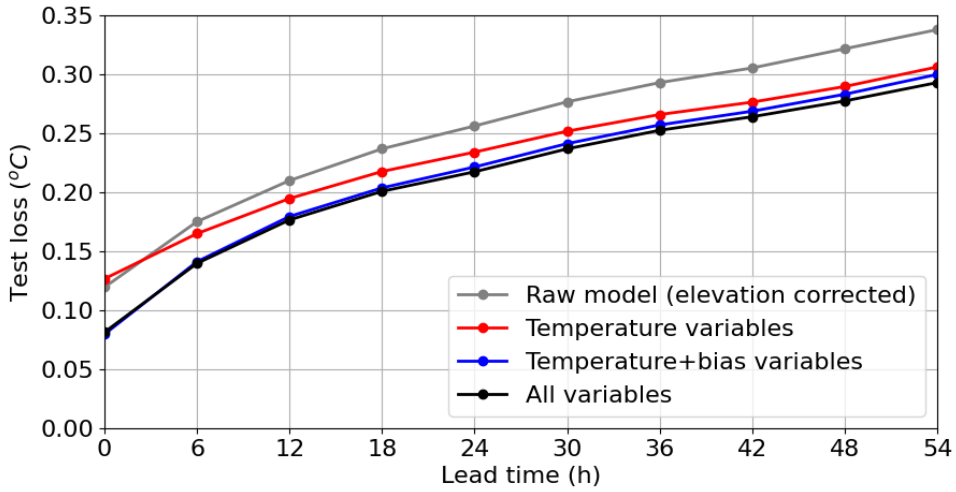


Fig. 3: Test loss against forecast lead time for models trained with varying sets of predictors.

3.1.5 Operationalization of model and future work

We have worked extensively on integrating the data processing pipeline and the U-Net model into MET Norway's operational forecasting system. We also performed a long training run using all available data (by merging the training and testing datasets) to create a trained model that we will use in production. The operational code is working and we produced a first test forecast (Fig. 4).

The next step is to perform some internal evaluation of the forecasts at MET-Norway, followed by a launch of the improved product on our weather app Yr (https://www.yr.no).

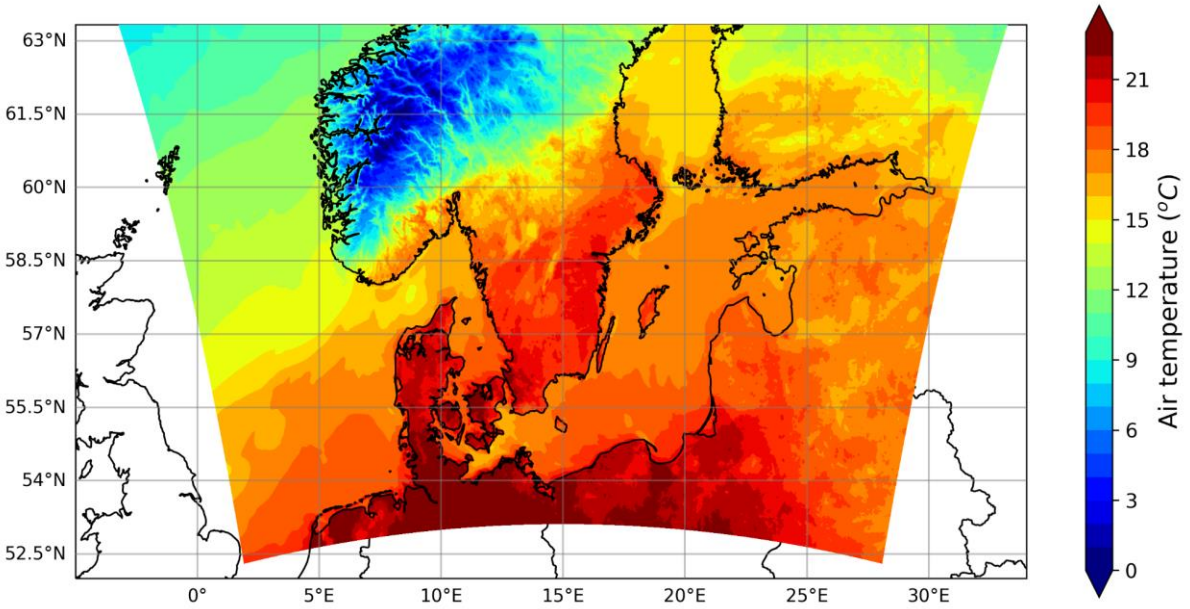


Fig. 4: An example forecast for 13:00Z on September 16, 2023 made using the final U-Net model. The map shows the 50th percentile forecast for the southern half of the domain.

3.1.6 Data and Code access

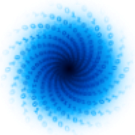
The training and testing code is available on https://github.com/metno/maelstrom-train, as in previous deliverables. The JUBE configuration for the scientific tests is found in jube/d1-4.yml.

3.2 AP2: Incorporate social media data into the prediction framework

Posts ons social media may provide relevant information about the current state of the weather at the location of the user. This application aims to harvest this information to improve weather predictions. For this, information related to weather needs to be extracted from a rather unstructured and unreliable data source. As a test balloon, we are developing a model that can determine from the text of a Tweet whether it was raining at the location and time at which the Tweet was sent. Our initial implementation was presented in Deliverable 1.3 (D1.3). Since then, we mainly focused on improving our training dataset. In addition, we added a new dataset based on precipitation data collected from weather stations, which improves our evaluation robustness. Finally, we give an outlook to planned improvements to our model.

3.2.1 Dataset

We use historical English Tweets from 2017-2020 that include keywords related to the presence of rain, e.g., “rain”, “sunny” or “drizzle.” We focus on Tweets sent from an identifiable location in the UK, which is required to map the Tweet to the precipitation dataset (see D1.3 for more details). Tweets may be linked to various geographical regions (tagged), e.g. “England”, “London” and/or “The British



Museum". To confidently map Tweets to precipitation values, we require the smallest tagged region to have an area at the resolution of the precipitation data (100 km²).

When trying to improve on our initial model, we realised that a major bottleneck for the model is data quality. Some Tweets clearly stated that it was raining when the precipitation data implied that it was not raining and vice versa. We therefore decided to move to ERA5-land (see below) as it incorporates measurement values to boost accuracy.

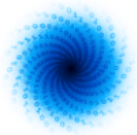
In addition, we build a dataset based purely on precipitation measurements from weather stations. For this, we use MIDAS Open (Met Office 2019), which provides hourly weather measurements from public weather stations in the UK from 1853 to present. We only include Tweets within 1 km of the weather station to foster data quality. This leaves us 34K Tweets with 4K Tweets labelled as "raining", which is insufficient to train our model. However, we use this dataset as a final holdout dataset to evaluate our model performance.

Precipitation data were assigned to the nearest Tweets. Tweets were assigned the label "raining" if total precipitation p was larger than a threshold value p_{thresh}, i.e. $p > p_{thresh}$. We set $p_{thresh} = 0.00\,7\,m$ as default. In our previous model, the threshold was set at the noise level of the simulation, which is much lower. However, further evaluation showed that we clearly over-estimated the presence of rain with this approach as the weather model will have larger inaccuracies at these low levels of rain. In addition, depending on the region users will probably be more sensitive to the presence of rain and a faint drizzle even if correctly predicted will not necessarily be considered "raining" by most users.

For precipitation data, we originally used ECMWF-IFS, but now moved to ERA5-land (Muñoz Sabater 2019), which has a spatial resolution of 0.1 deg and hourly resolution, which is comparable to the resolution of ECMW-IFS. However ERA5-land is a re-analysis dataset, which incorporates measurements to boost accuracy. The resulting training dataset contains 500K Tweets labelled as "raining" and 700K Tweets labelled as "not raining".

3.2.2 Model

Our task is to predict the labels "raining" or "not raining" from the text of Tweets. This can be reduced to the task of sequence classification, which is a common problem in natural language processing. Deep learning models based on the popular transformer architecture (Vaswani et al., 2017) deliver state of the art results in this domain (Kowsari et al. 2019). We use the transformer based DeBERTa architecture. Here, we use the pre-trained variant DeBERTaV3_small (see D1.2 for further details). As our data quality seemed to be the main bottleneck, we kept our model architecture unchanged and rather focused on improving our dataset and model evaluation. However, due to the benchmarks performed in D3.6, we noticed that while scaling on 4 GPUs compared to 1 GPU is less than ideal, energy consumption is still more efficient as the remaining 3 GPUs still require significant power in their idle state. We therefore always train on 4 GPUs now. In addition, we realised that while evaluation only takes 30 seconds for 50k Tweets, changing to a different architecture like a large language model (LLM) based architecture may incur considerable bottlenecks when running



predictions. Additional analysis performed for D3.6 gave us confidence in the performance of our model as it appears highly optimised, which allowed us to focus more time on improving our dataset.

3.2.3 Results

Our best model achieves an f1-score on the evaluation set based on ERA5-land of 0.74 (“not raining”) and 0.64 (“raining”). This corresponds to an AUC of 0.78, which corresponds to a slight improvement to our previous model (Fig. 5). It appears that the increase in accuracy of precipitation for our new dataset improved model performance. However, it is apparently not the main bottleneck for the model. Further analysis revealed instead that the information contained in a significant fraction of Tweets is not sufficient for a human to correctly classify these Tweets. Just matching by keywords appears therefore insufficient for selecting relevant Tweets.

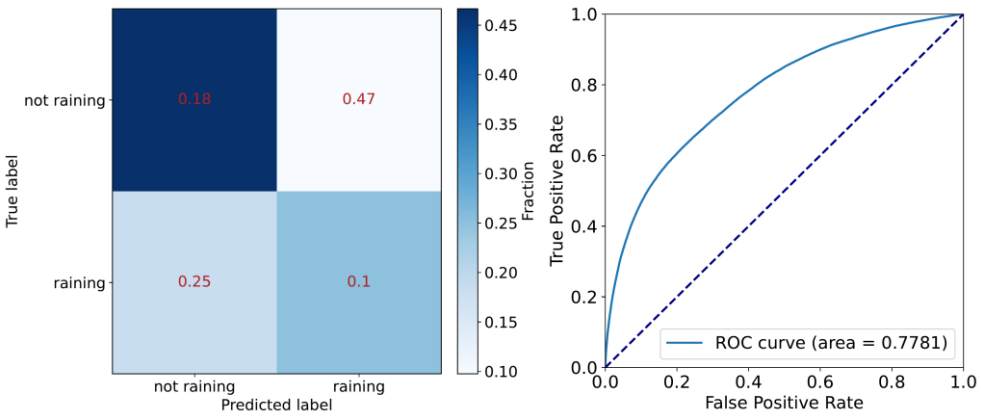
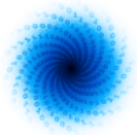


Fig. 5: Left: Confusion matrix of our best model for the classification of Tweets as “raining” or “not raining”. Right: ROC curve for the same model. While the majority of Tweets is correctly classified, further analysis on misclassified Tweets identified the lack of relevant information in the text to make a correct judgement as the main source of error.

3.2.4 Future work

A major challenge for this application is selecting Tweets that contain sufficient information to make the statement about the presence of rain or generally about the state of the weather. Currently, we filter out Tweets based on keywords. However, keywords can be ambiguous like the British newspaper Sun, which led us to introduce hand-crafted rules to filter out Tweets with the case-sensitive keyword “Sun”, which is rather incomplete. Instead, we would like to use a model that predicts the relevance of the Tweet for our specific task.

Large language models like ChatGPT (OpenAI 2023) or alternative open source solutions like Falcon 180B (Schmid et al. 2023) provide a new paradigm in the NLP community. Initial manual testing gave promising results that demonstrated that LLMs can potentially help with this complex task. However, a major challenge appears to be speeding up the evaluation process as potentially a stream of Tweets needs to be processed from which relevant Tweets are selected. We therefore plan to finetune an LLM specifically for our task, which should improve performance.



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

AP3 concerns the emulation of radiative transfer with machine learning. The motivation is the acceleration through both compression via machine learning and easy use of GPU infrastructure. In previous deliverables, we focussed on shortwave (SW) radiation, where the sun is the primary source. We designed a bespoke physics-informed neural network whose architecture mimicked the conventional physics-based solver. This network belongs to the family of recurrent neural networks (RNN), here recurrent through the vertical structure of the atmosphere. Leveraging this solution, and training on the large dataset provided through MAELSTROM, we were able to train highly accurate emulators of the ECMWF ecRAD Tripleclouds solver for the shortwave process. These exhibited RMSE errors of 0.50 W/m^2 for the fluxes and 0.014 K/d for the heating rates on an independent in time test dataset. In consultation with domain experts, it was decided that this error was sufficiently low to consider the emulation successful, at least from the perspective of offline errors, i.e. errors when running the radiative transfer solver decoupled from the weather forecasting model.

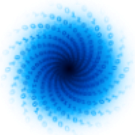
Our focus in the time since the last deliverable has been on the longwave (LW) process for radiative transfer. In combination with shortwave radiation processes, this complements the radiative transfer task. Longwave emissions primarily originate at the surface, with it being key to identify: how much is reflected back from the atmosphere to be reabsorbed by the surface, how much is emitted and absorbed by the atmosphere, and how much flux propagates through the top of the atmosphere.

3.3.1 Data

For this phase of reporting, we continue using the same dataset as described previously, as this was found to be sufficiently accurate for learning the shortwave process. The dataset contains data for both shortwave and longwave processes. These can be accessed from the `maelstrom-radiation-tf` CliMetLab dataset using the `subset = "tier-2"` argument. The corresponding dataset comprises 21,640,960 examples. This is to be contrasted with the Tier 1 dataset which contained only 67,840 examples. We also introduced previously 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). Here, we will present results on the Tier 2 data.

3.3.2 Methodology

Given the success of recurrent neural networks for the shortwave process, we focus our attention on adapting this solution to the new problem. As the dominant source is the surface, we change the structure of the RNNs. Scalar surface quantities are passed to a 2-layer multi-layer perceptron (MLP) enabling the model to process surface quantities. The output of this MLP is provided as an input state to an RNN which is oriented to propagate vertically up the atmospheric column. At each layer, the RNN receives the previous layers' hidden state and the forcing is provided by atmospheric state variables corresponding to this layer of the atmosphere. This RNN propagates information from the



surface up through the atmospheric column, incorporating information on the state of the atmosphere (e.g. temperature, humidity pressure) at each layer. This is followed by a second RNN, which propagates down the atmospheric column, initialised with the final hidden state of the upward-propagating and is forced with the output state from the upwards RNN at each layer. The second RNN enables the propagation of reflectances back down the column all the way to the surface. The output at each layer from both RNNs is concatenated together and passed through a single MLP, with shared weights for all vertical layers, which is applied to each layer separately and calculates a normalised flux profile (i.e. two components downwards and upwards fluxes). For the SW process, fluxes at the end of the model are scaled by the incoming solar radiation, which simplifies the task for the rest of the network to only calculating relative fluxes. We adopt this approach for LW, but here the scaling factor is not known a-priori, but depends on surface properties. We use a 2-layer MLP to calculate this scaling factor and multiply the downwards and upwards flux profiles by this prediction. Finally, again based on the success from the SW application, we use the custom physics-informed layer which calculates the heating rates from the fluxes. This final step does not add further trainable parameters and the formulation can be found in D1.3.

We test two different RNN blocks, the simpler Gated Recurrent Unit (GRU) and the more complex Long Short-Term Memory (LSTM). We fix a hidden state of 64 neurons for each MLP and RNN in the model, having found this to be sufficiently large on preliminary testing. The GRU-based model has 45,221 trainable parameters and the LSTM-based model has 60,261 trainable parameters. As with the shortwave process, we produce a highly parameter efficient model in contrast to recent publications using only MLP or convolutional neural networks which use on the order of million parameters (Lagerquist et al. 2021, Yao et al. 2023).

Exploratory testing was also carried out to test the suitability of other architectures for learning accurate emulators. Specifically we tested convolutional neural networks, using dilated convolutions to increase the speed of propagation through the vertical column. Also tested were networks using the self-attention mechanism that powers transformer architectures. None of the probed models attained the accuracy of the RNNs as shown in Figure 6.

We minimise the Huber loss, which acts as the mean-absolute-error for values greater than 1 and as the mean-squared-error for values less than 1. This was found to result in more accurate models, measured by RMSE, even compared with training directly on RMSE. To combine the loss of two objectives, we use a weighting of 1 for the fluxes and 100 for the heating rates.

Training was carried out with a batch size of 512, the Adam optimiser, an initial learning rate of 0.001 which decreases if validation scores fail to improve after 4 epochs. Early stopping if scores on the validation dataset failed to improve after 6 epochs was also used. Training was carried out on a single NVIDIA A100-40Gb GPU, since larger batch sizes using data-distributed parallel training did not improve the time-to-solution for the application at hand as shown in previous deliverables. An optimal solution was reached in approximately 24 hours.

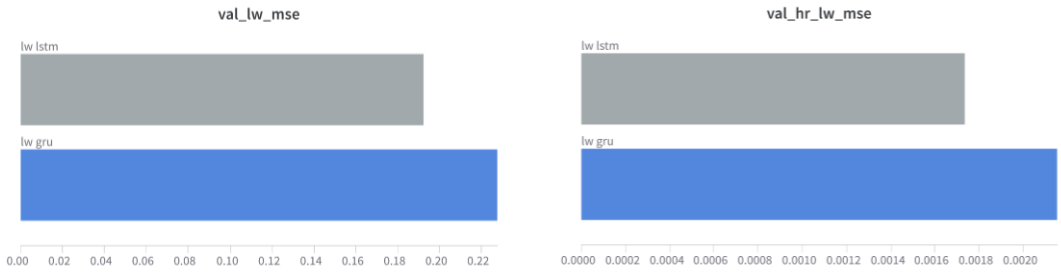


Fig. 6: Mean-squared-error (MSE) performance on test dataset for longwave fluxes (left) and longwave heating-rates (right), comparing model designs featuring GRU-type RNNs and LSTM-type RNNs. LSTMs reduced errors for both fluxes and heating rates by approximately 10%.

3.3.3 Results

Compared to the shortwave solver, where our leading solution attained an RMSE of 0.50 W/m^2 for the fluxes and 0.014 K/d for the heating rates, we now see that estimating the fluxes results in slightly smaller errors (0.43 W/m^2), whereas the heating rate errors are larger (0.042 K/d). This is consistent with challenges in the dataset, as the magnitude of incoming shortwave radiation is significantly larger than for longwave, whereas the heating rate profiles for longwave radiation are more complex in their vertical structure. This is demonstrated in the plots below which show sample predictions for shortwave and longwave processes (Fig. 7 and 8).

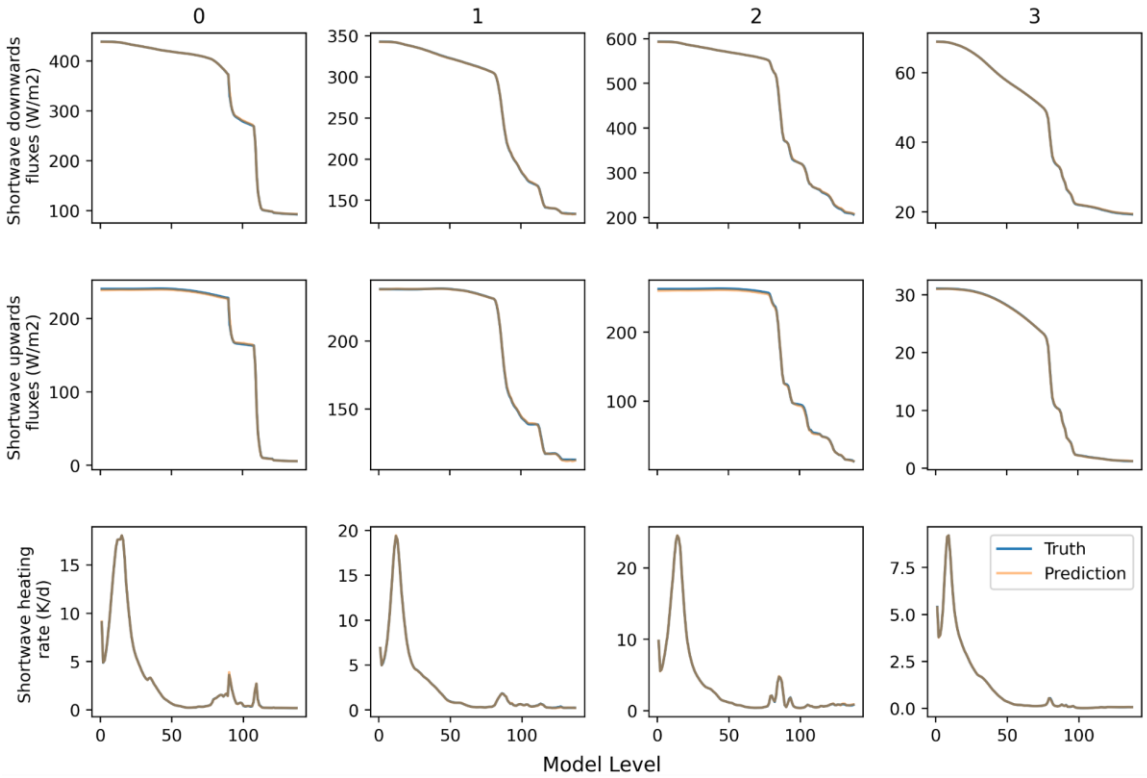


Fig. 7 : Sample profiles of predictions from the leading SW RNN model. Columns indicate different profiles, with

rows corresponding to the downwards flux, upwards flux and resulting heating rate. Blue and orange lines coincide almost everywhere, illustrating the high quality of prediction.

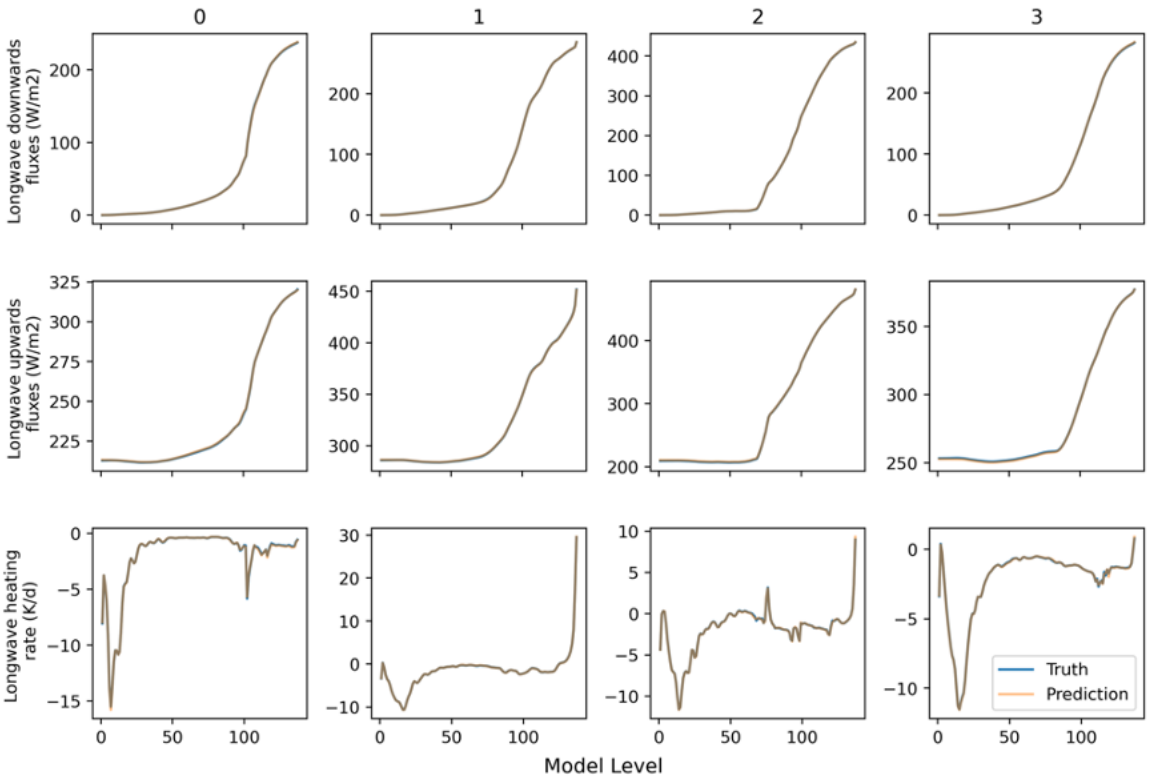


Fig. 8: Sample profiles of predictions from the leading LW RNN model. Columns indicate different profiles, with rows corresponding to the downwards flux, upwards flux and resulting heating rate. Blue and orange lines coincide almost everywhere, illustrating the high quality of prediction.

3.3.4 Code

The code for the longwave training has been added to the benchmark code in the MAELSTROM Radiation repository, https://git.ecmwf.int/projects/MLFET/repos/maelstrom-radiation. Once pip installed, training can be invoked from the command line with

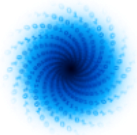
```
radiation-benchmarks-lw
```

which runs an example on the tier 1 dataset. Training for the optimal longwave model can be achieved with

```
radiation-benchmarks-lw --epochs 100 --tier 2 --batch 512
```

3.3.5 Future work

As with the shortwave process, we believe that these errors are low enough to satisfy offline testing. Over the remainder of the project we will focus on two aspects: Firstly, online testing, i.e. coupling both solvers simultaneously to the IFS and testing forecast accuracy and stability. Secondly, generating tangent-linear and adjoint versions of the shortwave and longwave processes. These gradient versions



of the neural network have value within data assimilation, particularly variational approaches such as 4D-var which seeks to optimise forecast trajectories by propagating gradients through the full forecasting system. This requires accurate and fast gradient models for each component, which necessitates significant personnel investment to develop and maintain. Currently, the tangent-linear and adjoint models for the physical radiative transfer scheme in ECMWF's IFS correspond to an older version of the radiation scheme. If the gradient information provided by an automatically differentiated neural network version of the radiative transfer process is sufficiently accurate, this would improve the correspondence between forward and gradient models and could significantly improve initial conditions and resulting weather forecasts.

3.4 AP4: Improve ensemble predictions in forecast post-processing

Over the past few years, there has been a growing interest in ensemble post-processing techniques aimed at enhancing forecast accuracy. These methods involve adjusting the distribution of output from ensemble weather prediction models to eliminate biases, a process known as prediction correction. They play a crucial role in enhancing the overall quality of ensemble forecasts.

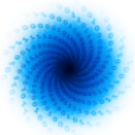
The primary objective of AP4 is to enhance the performance and reliability of machine learning-based models within the context of ensemble post-processing, also known as forecast post-processing. In pursuit of this goal, we have taken the initiative to introduce the ENS-10 (Ashkboos et al., 2022) dataset. This dataset comprises ten ensemble members, encompassing a time span of two decades, from 1998 to 2017. Our approach involves the application of various machine learning models to this extensive dataset, with the aim of advancing the state-of-the-art in ensemble post-processing techniques

3.4.1 I/O Bottleneck

In our previous deliverable, D1.3, we conducted an assessment of the effectiveness of implementing a U-Net style model on the ENS-10 dataset. In this evaluation, we carefully analysed the time allocation between forward and backpropagation processes for each batch and epoch. Our findings indicated a significant bottleneck in the training process due to input/output (I/O) operations. Therefore, it is evident that any future efforts aimed at enhancing performance should prioritise optimization at this particular stage of the application.

This deliverable primarily focuses on optimising the overall runtime performance of our model by addressing a critical aspect: input/output (I/O) operations. As part of our investigation, we have recognized that the current practice of saving our dataset in *NetCDF* format and conducting computations during the data loading phase introduces a substantial I/O overhead that significantly impacts the efficiency of our neural network training process.

To elaborate, our focus is on streamlining the end-to-end execution of our model, encompassing data handling and neural network training. A key observation is that the choice of saving the dataset in the NetCDF format, while beneficial for structured data storage, brings with it the challenge of I/O efficiency during the training phase. When we perform computations on-the-fly during data loading, it results in increased reading times and consequently slows down the overall training process.

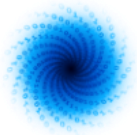


3.4.2 NumPy and NetCDF Formats

Table 3 illustrates the distinctions between the utilisation of NetCDF and NumPy formats. In our efforts to address the I/O challenges, we are contemplating a potential solution: preprocessing the entire dataset by performing all necessary computations, such as normalisation and slicing, and subsequently storing the resulting data in NumPy format.

Use Case	NetCDF Format	NumPy Format
Data Conversion Ease (to PyTorch Tensors)	Requires additional steps for data conversion. PyTorch does not have built-in support for direct conversion from NetCDF files, necessitating custom code or external libraries for the conversion.	Offers straightforward conversion as PyTorch seamlessly supports direct conversion from NumPy arrays using *torch.from_numpy()*.
Conversion Speed and Efficiency	Conversion from NetCDF to PyTorch tensors may involve reading and copying data, potentially resulting in increased conversion times, especially for large datasets.	Conversion from NumPy arrays to PyTorch tensors is generally efficient, with minimal overhead, making it suitable for quick data loading and model training.
Memory Usage	NetCDF format may involve reading data into memory, potentially consuming more memory during conversion, especially for large datasets.	NumPy arrays are memory-efficient, and PyTorch can efficiently create tensors from NumPy arrays without significant memory overhead.
Compression and File Size Optimization	Offers built-in compression options for reducing file sizes while preserving data integrity.	Compression can be applied externally but requires more manual configuration compared to NetCDF's built-in compression.
Data Complexity	Ideal for complex data with numerous dimensions and associated metadata, such as climate data, geospatial data, and model outputs.	Suitable for a wide range of numerical data but may lack the structure needed for complex, multidimensional data.

Table 3: Comparison between data provision using the NetCDF and NumPy format. We fix our I/O issue by saving the whole ENS-10 dataset in NumPy format and applying the normalizations offline.



3.4.3 Results

To assess the effectiveness of our I/O enhancements, we conducted training experiments utilising the U-Net network architecture as described in the work by Ashkboos et al. (2022) on the ENS-10 dataset. Our U-Net model consists of three hierarchical levels, each comprising a sequence of modules, including convolution, batch normalisation, and ReLU activation functions.

For data input, the network operates on the entire grid and processes data with 22 input dimensions for surface features and 14 input dimensions for volumetric features. This corresponds to two inputs per variable present in the ENS-10 dataset, facilitating comprehensive data representation. In terms of network architecture, we employed convolutional layers with 32 output channels in the first level, 64 output channels in the second level, and 128 output channels in the third level of our U-Net model.

For all experiments, we use a single 40 GB A100 GPU from ETH computing resources using CUDA 11.7 and train our model for three epochs. We use PyTorch 2.0.0 to implement our code and run the experiments. We train each model using three different random seeds and report the mean and standard deviation. Batch, forward, and backward time was measured using GPU-side timing.

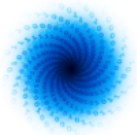
Metric	Min(s)	Max(s)	Median(s)	Mean(s)	Std(s)
Epoch	1098.97	1102.71	1101.08	1100.92	1.87
Batch	1058.43	1062.31	1061.08	1060.61	1.98
Forward	26.91	27.94	27.92	27.59	0.59
Backward	12.47	13.09	12.60	12.72	0.32

Table 4: The Benchmarking results for AP4 using NetCDF format.

Metric	Min(s)	Max(s)	Median(s)	Mean(s)	Std(s)
Epoch	214.75	226.27	220.20	220.40	5.76
Batch	172.11	183.01	177.28	177.47	5.45
Forward	30.31	30.42	30.33	30.36	0.057
Backward	12.323	12.84	12.56	12.59	0.25

Table 5: The Benchmarking results for AP4 using NumPy format.

Tables 4 and 5 show the results of our experiments using the NetCDF and NumPy formats. We get up to the **4.9x** speed-up by using NumPy data format in our experiments. Specifically, the mean time of each epoch is reduced from 1100.92s to 220.4s.



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.
So far, the statistical downscaling models have been developed in an end-to-end approach. This means that task-specific neural networks have been trained from scratch using pairs of coarse-grained input and high-resolved target data. However, pairing of input and target data often limits the amount of training samples since both datasets must cover the same time period and region for this purpose. In the case of the Tier 2-dataset, the ERA5-reanalysis, which serves as the coarse-grained input, provides global data from 1979 until near real-time. By contrast, the high-resolved targeted COSMO REA6-reanalysis constitutes a regional dataset whose temporal coverage is restricted from Jan 1995 to August 2019. Thus, large parts of the ERA5-reanalysis dataset cannot be leveraged during training in an end-to-end approach.

Inspired by the recent success of foundation models in natural language processing (see, e.g., Zhou et al., 2023), we therefore explore the applicability of the large-scale representation model for atmospheric dynamics AtmoRep (Lessig et al., 2023) in scope of this deliverable. AtmoRep constitutes a task-agnostic generative neural network that is considered to be suitable for a wide range of meteorological applications and thus represents a foundation model (Bommasani et al., 2021). By pretraining the encoder-decoder transformer network of AtmoRep on (nearly) the complete ERA5 reanalysis dataset, a powerful abstraction of the atmospheric state can be obtained. This abstraction can then be exploited when finetuning a task-specific network extension (encoder-decoder of AtmoRep +tail network) for statistical downscaling.

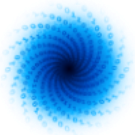
A compact overview on AtmoRep and its application for statistical downscaling is provided in the following. For a detailed description of AtmoRep, we refer however to Lessig et al. (2023) and its comprehensive supplement.

3.5.1 The AtmoRep model and fine-tuning for downscaling

Mathematically, AtmoRep is built on the description of the atmospheric state as a stochastic dynamical model (see, e.g., Hasselmann, 1976 and Palmer et al., 2008). With this, the probability for a state y given the input atmospheric state $\vec{x}$ is described as a conditional probability distribution $p(y|x,\alpha)$. The input state x can, for instance, be the atmospheric state at time t_0, whereas y represents the atmospheric state at a later time $t = t_0 + \Delta t$. The atmospheric state is here complemented by α which may provide auxiliary information such as the year information to encode global climate forcing. Since no analytical description of the highly complex and non- stationary stochastic system is available, AtmoRep makes use of the approximation

$$p_\theta(\hat{y}|\hat{x},\hat{\alpha}) = p(y|x,\alpha)$$

where $p_\theta(\hat{y}|\hat{x},\hat{\alpha})$ constitutes an encoder-decoder neural network based on Transformer blocks (Vaswani et al., 2017). As already mentioned, the ERA5 reanalysis dataset serves as input data, providing the most accurate, available estimate on the global atmospheric state. Specifically, different state variables such as the 3D wind-vector, the temperature or the specific humidity on different model levels are inputted in the form of gridded data within local space-time cubes (e.g. 36h x 5 vertical levels x 1800 km x 1800 km). To learn an abstract representation of the atmospheric dynamics,



the data cubes are tiled into patches in analogy to tokens in computer vision (Dosovitskiy et al., 2020). During training, a random subset of the tokens gets masked or distorted and the encoder-decoder network is asked to reconstruct these tokens. This strategy of self-supervised training is inspired by the BERT training strategy suggested in Devlin et al., 2018. Here, high masking ratios have been found to promote learning of robust representations.

AtmoRep thereby constitutes a probabilistic model by outputting an ensemble for the intrinsically uncertain dependant state $\hat{y}$. A novel statistical ensemble loss that measures the distance between the (deterministic) ground truth data and the probabilistic prediction with the help of a Gaussian fit is used as a training objective combined with the usual MSE-loss. A conceptual illustration of AtmoRep is provided in Figure 9.

For efficiency reasons and for the sake of a modular model design, the pre-training is performed as a two-step approach. At first, state variable-specific encoder-decoder networks (Singleformers) are trained individually until convergence, i.e. $\hat{x}$ and $\hat{y}$ are just expressed by a single variable on multiple levels in the Singleformers. Second, different variables are coupled together via cross-attention between the variable-specific Transformer layers of the encoder and training of the resulting Multiformer is continued to obtain a more complete abstraction of the atmospheric dynamics. This approach has the advantage of saving computation time, since the optimization converges quickly in the latter step, while the cross-attention operations scale quadratically as opposed to the self-attentions. Furthermore, task-specific configurations with different variables become possible through the two-step training approach and the modular design.

For the downscaling task, the horizontal wind vector components as well as the temperature on model levels 137, 123, 114, 105 and 96 are used since this information is considered to be the most relevant for the subsequent downscaling of the 2m temperature field. The Singleformers of the three variables have been trained individually on 8 nodes (32 GPUs) of Juwels Booster for several days. Subsequently, the Multiformer is trained, before AtmoRep gets extended with a task-specific tail network for downscaling. The tail network comprises 6 transformer blocks consisting of transformer layers with 16 attention heads and two multilayer perceptrons. To increase the spatial resolution of the data, the output tokens of AtmoRep's decoder get increased by a factor of 4 together. The increased number of tokens also require embedding (linear layer) with an updated local positional encoding, while the embedding dimension also gets doubled. In contrast to the three Singleformers and the Multiformer only data from model level 137 is used in the fine-tuning step.

During fine-tuning, the parameters of the encoder-decoder as well as the tail network are optimised, resulting in about 1.85 billion trainable parameters. Model parallelism is required to fit the network

on the computing nodes of Juwels Booster, where the three state variable-specific transformers and the tail network are placed on one GPU each. The finetuning is run for three days on eight nodes.

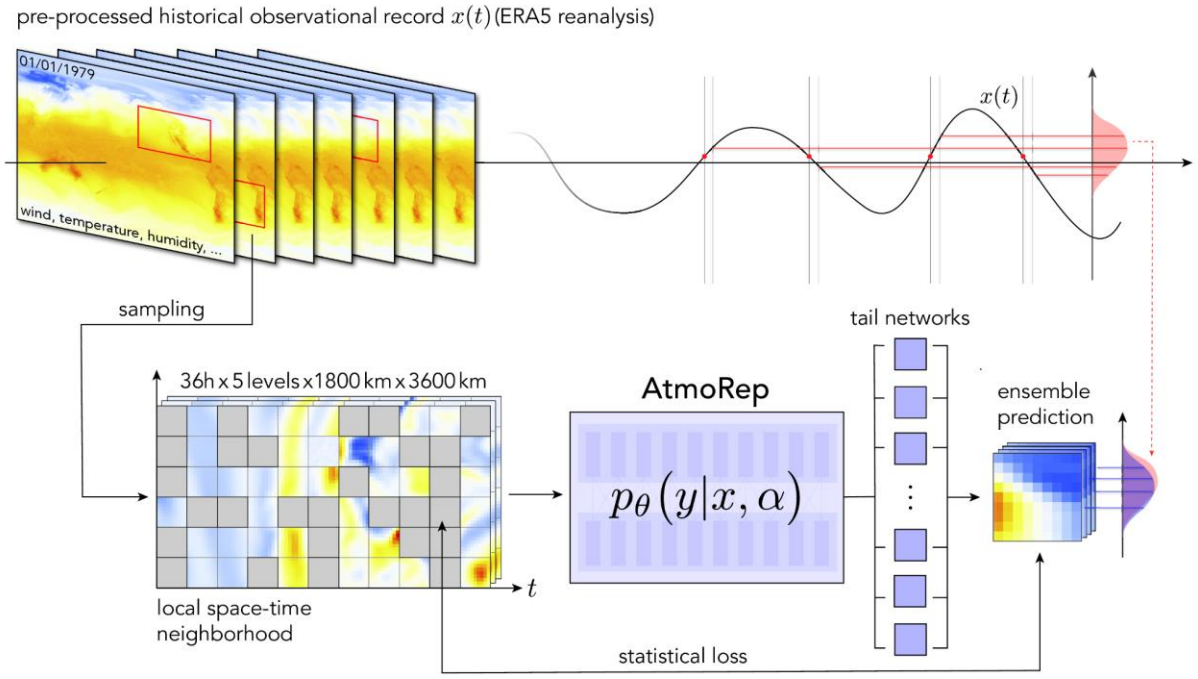


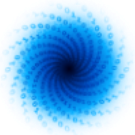
Fig. 9: Illustration of the self-supervised training of AtmoRep. Local space-time data cubes of state variables are sampled and then tailed into tokens of which a random subset gets masked. AtmoRep then reconstructs the data of the masked tokens with an ensemble prediction. The statistical loss is then used for optimization.

3.5.2 Dataset

While the pre-training of the Singleformers and the Multiformer are solely performed with the global ERA5-reanalysis dataset (from 1979 to 2017), the subsequent finetuning requires pairing with the COSMO REA6 dataset. Contrarily to the Tier-2 dataset used in previous deliverables, the underlying shared grid projection of the data is changed from the rotated pole grid of the COSMO REA6-dataset to the regular (lat, lon)-grid onto which the ERA5 reanalysis is provided. The input ERA5-data is defined on a 0.25°-grid, while the COSMO REA6-data is remapped on a 0.0625°-grid following the procedure described in deliverable 1.1.

Variable (variable name)	Model levels	Data Source (grid spacing)	Input/Output
temperature (t)	96***, 105***, 114***, 123***, 137***	ERA5 (0.25°)	Input
u-wind (u)	96**, 105**, 114**, 123**, 137***	ERA5 (0.25°)	Input
v-wind (v)	96**, 105**, 114**, 123**, 137***	ERA5 (0.25°)	Inout
surface geopotential (z)*	-	ERA5 (0.25°)	Input
2m temperature (t_2m)	-	COSMO REA6 (0.0625°)	Output
surface topography (hsurf)*	-	COSMO REA6 (0.0625°	Input/Output

*Table 6: Overview of input and output variables used for the 2m temperature downscaling task. Variables denoted with * served as auxiliary input/output variables for the competing WGAN. Data on model levels*



*denoted with ** have been used exclusively for pre-training AtmoRep, whereas *** represents data on model levels that has been inputted to the WGAN and to AtmoRep.*

For the finetuning of the downscaling task, a fixed spatial neighbourhood comprising 6x12 tokens in (lat, lon) direction is defined. Each token covers 9x9 grid points in (lat, lon)-space of the ERA5-data, which corresponds to 36x36 grid points for the target COSMO-REA6 data. Thus, the resulting domain comprises 216x432 grid points on the high-resolved 0.0625°-grid which is about 8x larger than the number of grid points in the Tier2-dataset. As displayed in Figure 10, the corresponding domain covers large parts of Central Europe including the Alps and parts of Eastern Europe.

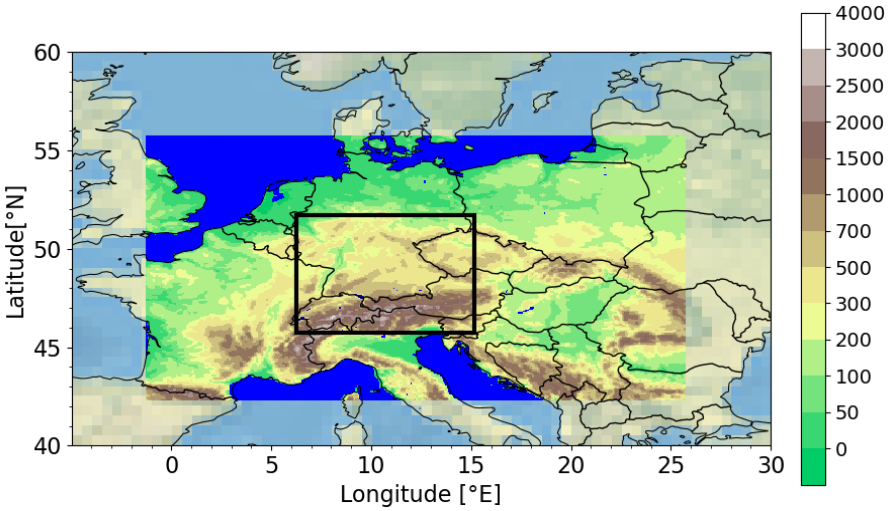


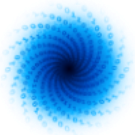
Fig. 10: Target domain of the AtmoRep downscaling application. The surface topography in metres above sea level is used to highlight the domain. The target domain of the trained competing WGAN-model is rendered in black.

The (labelled) training dataset for downscaling comprises all years from 1995 to 2017 from which four months have been omitted due to missing data samples in the COSMO REA6-dataset. With hourly data, more than 200K samples become available for training. The data for 2018 is held out for testing.

For the competing WGAN experiment (see below), a smaller target domain with 144x96 grid points is chosen. This domain size is similar to the one chosen for the Tier-2 dataset and was chosen due to efficiency reasons considering the spatial invariance property of the convolutional operators used in the WGAN architecture. The smaller sub-domain is also highlighted in Figure 10. To provide temperature information on elevated model levels to the competing WGAN that is also used for the pre-training of AtmoRep, the temperature at model levels 96, 105, 114 and 123 is provided in addition to the wind and temperature data on model level 137.

3.5.3 Experiments

In scope of this deliverable, the downscaling experiment with the AtmoRep configuration as described above is focussed. To provide a preliminary assessment on the added value of using AtmoRep, the results are compared to a WGAN experiment. The applied WGAN is the same as in D1.3, only the dataset changes as described in 3.5.2.



3.5.4 Results

In the following, the downscaling experiment with AtmoRep is evaluated in terms of the RMSE, the bias and the spatial variability. The latter is quantified by comparing the domain-averaged amplitude of the horizontal 2m temperature gradient between the downscaled and the ground truth data. In the following, the term gradient ratio will be used for this metric, whose optimal value is 1. Furthermore, a spectral analysis of the resulting 2m temperature field is performed.

3.5.4.1 Downscaling with AtmoRep

Figure 11 shows the diurnal cycle of the RMSE (a) and the gradient ratio (b) averaged over the complete test year 2018. The overall mean of the $RMSE_{AR}$ is 0.998 ± 0.151 K with smallest errors in the late evening and a weak peak around noon. This RMSE-value constitutes an improvement of about 18% (12%) compared to the WGAN (U-Net) tested in D1.3. Thus, a significant improvement can be deduced, albeit the underlying domains differ here. A more consistent comparison will therefore be undertaken below. The overall averaged bias is close to zero, but a slight positive bias of about 0.15 K is present around noon (not shown).

In terms of spatial variability however, the downscaling product of AtmoRep underestimates the spatial variability by about 15%. The obtained results are comparable to the U-Net model, whereas it is clearly inferior to the WGAN model evaluated in deliverable 1.3. In analogy to the gradient ratio, the overall averaged power spectrum in Figure 12 (left) also shows less spectral energy at high wavenumbers compared to the target COSMO REA6-data, indicating that the variability is most noticeably underestimated at the smallest spatial scales.

Inspecting the seasonality of evaluation scores furthermore reveals that the noon-peak in the RMSE is strongest in the summer months (Fig. 12 right). The strongest error contributions are hereby from the western and eastern parts of the domain rather than from the Alpine region. It is hypothesised that these are related to the pronounced positive 2m temperature anomalies of summer 2018. However, further investigation is required to underpin and explain this assumption.

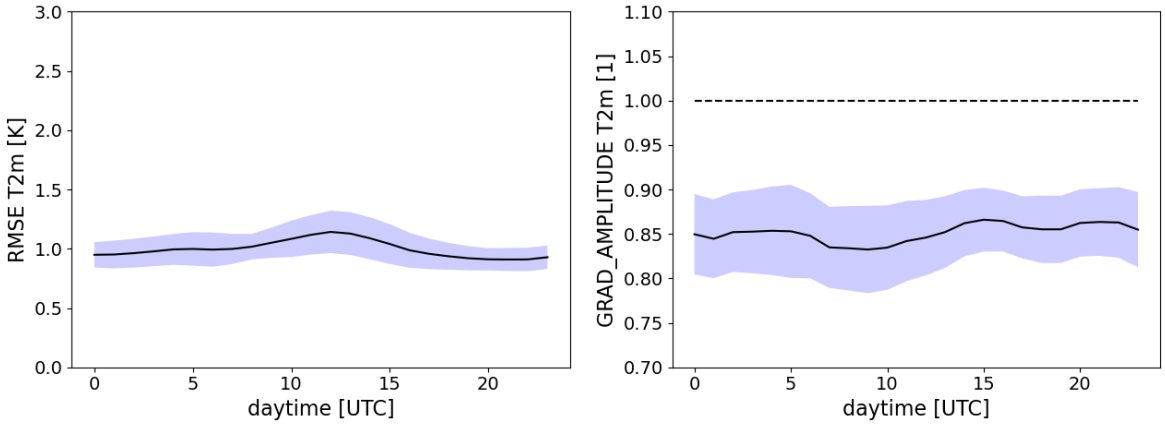


Fig. 11: Diurnal cycle of the RMSE (left) and the gradient ratio (right) as averaged over the test year 2018 from the AtmoRep downscaling experiment. The shaded area represents the standard deviation of the scores as an estimate of their uncertainty.

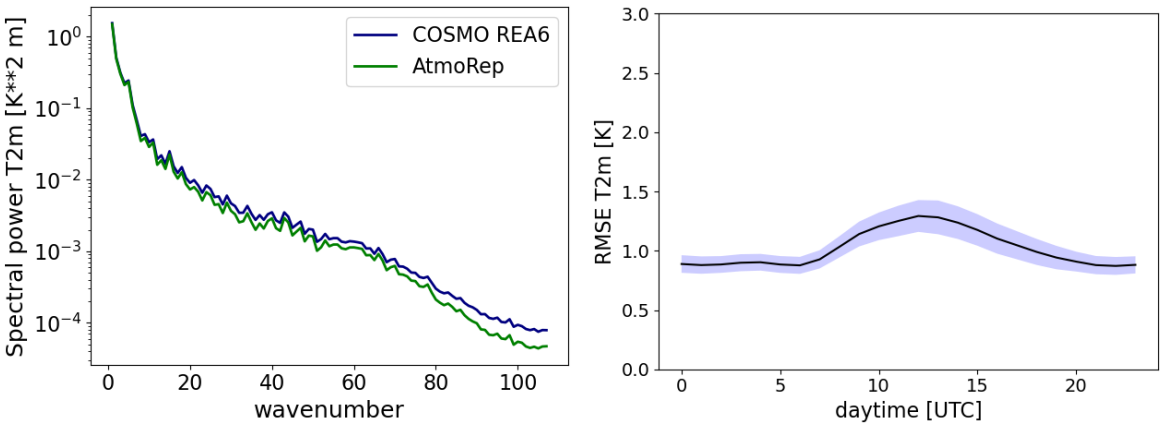


Fig. 12: Left: Power spectrum of the 2m temperature field from the COSMO REA6 ground-truth data (blue line) and the downscaled product of AtmoRep (green line). The spectra are averaged over the whole test year. Right: Diurnal cycle of the RMSE for the summer months JJA 2018.

3.5.4.2 Comparison with WGAN

The comparison results on the consistent domain (cf. Fig. 10) between downscaling with AtmoRep and the WGAN model (cf. Subsection 3.5.3) are presented in Figure 13. In terms of the RMSE (Fig. 13 left), AtmoRep ($RMSE_{AR} = 0.989 \pm 0.215$ K) clearly outperforms the WGAN ($RMSE_{WGAN} = 1.163 \pm 0.319$ K). The errors are smaller for all daytimes with the strongest differences in the afternoon and evening hours. At this time, the WGAN shows up with its largest RMSE-values, whereas AtmoRep starts to obtain its minimum values. In general, the daytime dependency is much larger for the WGAN which can be attributed to the efficient time-embedding (part of $\hat{\alpha}$) in AtmoRep.

However, in terms of the spatial variability, the WGAN clearly outperforms AtmoRep (Fig. 13 right). In accordance with the results presented in D1.3, the adversarial training of the WGAN constitutes an efficient way to obtain the desired spatial variability in the downscaling product, even though the spatial variability is slightly underestimated (overestimated) in the morning (afternoon) hours. By contrast, AtmoRep produces results that are similar to the U-Net as presented in D1.3. However, it is also worth mentioning that the underestimation in the spatial variability is partly due to the averaging over the four ensemble members of AtmoRep's probabilistic output. Investigating individual ensemble members reveal a slightly higher degree of spatial variability, albeit resulting in a small increase of the RMSE as well.

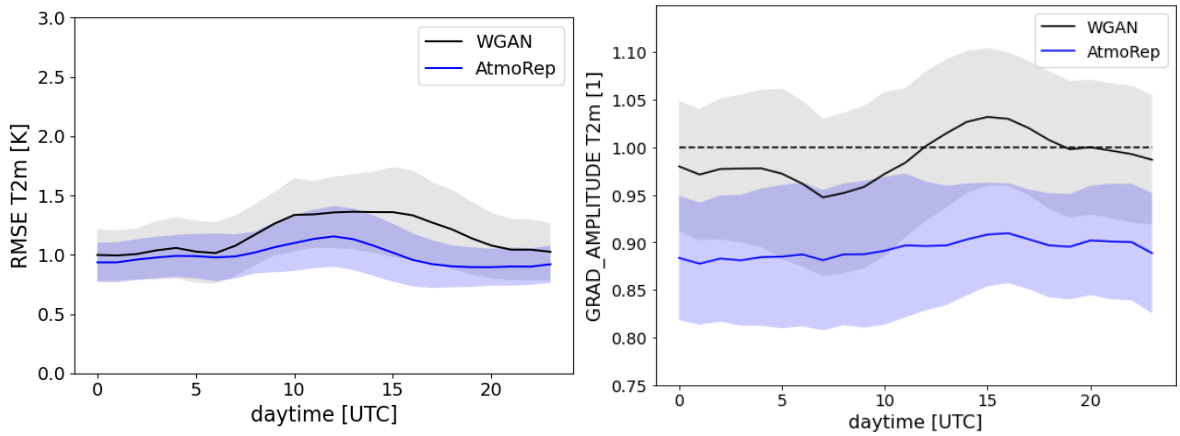


Fig. 13: As Figure 11, but joint evaluation of results obtained with AtmoRep (blue) and the WGAN (black).

3.5.5 Discussion and Outlook

Downscaling with the large-scale representation model AtmoRep has the potential to significantly improve the accuracy of the downscaled data. The experiment conducted in scope of this deliverable revealed a significant reduction in terms of the RMSE compared to the WGAN model which has been probed in previous deliverables. This is not only true for the case when the same predictor variables are provided to the WGAN, but also when more predictors are provided. However, in terms of the spatial variability, AtmoRep is still clearly inferior. The amplitude of the local temperature gradient is clearly underestimated, albeit the underestimation is smaller for individual ensemble members.

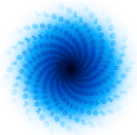
Future work of downscaling with AtmoRep is therefore dedicated to also increase its performance in terms of the spatial variability. One direct approach is to increase the number of ensemble members to allow for a better representation of the intrinsic uncertainty in the downscaled output. Yet, an ensemble member size of 4 is considered to be insufficient for such a desired property of AtmoRep's intrinsically probabilistic framework.

However, increasing the number of ensemble members requires a more efficient model configuration during finetuning. In scope of this deliverable, the encoder and decoder of AtmoRep have been kept trainable resulting in an enormous amount of trainable parameters, thereby requiring for model parallelism. In the future, more lightweight model configurations during finetuning will therefore be probed, e.g. by freezing the parameters of the pretrained encoder.

This will not only allow for a larger number of ensemble members, but will also enable adding more predictors such as surface heat fluxes or the high-resolution topography. The latter is considered to further boost the accuracy in terms of the RMSE, but also the spatial variability.

3.5.6 Data and code access

The code to run the competing downscaling model can be accessed in the project repository on gitlab . Instructions to run the training as well as the postprocessing are provided in the README. The corresponding training data and a tar-archive of the AtmoRep downscaling output can be downloaded via the CliMetLab plugin.



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

Power production forecasts for renewables based on data produced by numerical weather prediction (NWP) are prone to uncertainty. The uncertainty of forecasts is only quantifiable in retrospect, though. Operators of wind or solar parks who follow regulations of national energy markets or actively participate in energy trading, however, require an estimate for the likely uncertainty of power forecasts.

AP6 targets to find solutions that allow the quantification of forecasts a priori. Large-scale weather regimes (LSWRs) may be the key: if the uncertainty of forecasts correlates with the presence of specific LSWRs in the area of a park and is quantifiable, identification of LSWRs from NWP predictions would support the uncertainty estimation of the delivered power production product. This would be a huge benefit to operators since it would give them confidence for their measures taken based on a forecast.

To achieve classification of LSWRs, we make use of the DeepClusterV2 (DCv2, Caron et al. 2020) algorithm: a deep-learning approach to unsupervised clustering of image data. DCv2 uses a ResNet50 architecture feeding a multi-layer perceptron (MLP) followed by a spherical K-Means clustering to produce pseudo-labels (classes) to use for back propagation using the Cross-Entropy Loss (Fig. 14). To increase the robustness of the result, DCv2 employs data augmentation by training the ResNet50 and MLP with multiple random crops of the input data.

As input, we use the data from D1.2, which consists of data from the ECMWF IFS HRES model from 2017–2020 with a spatial resolution of 0.1° and temporal resolution of 1 hour. For each sample, we calculate the daily mean and create pseudo-RGB images by using absolute wind speed, temperature, and relative humidity for each channel. We train the model for 800 epochs with an adaptive learning rate using Large Batch Training of Convolutional Networks (You et al. 2017). For the K-Means, we choose $K = 30$.

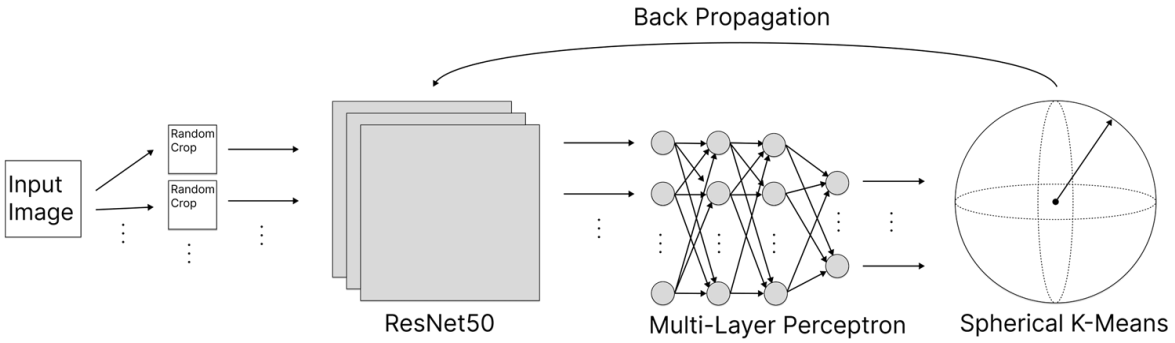
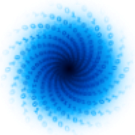


Fig. 14: Illustration of the DCv2 algorithm. Input to the models is a certain number of random crops of the original input samples. The ResNet50 and MLP are trained with the pseudo-labels assigned to each sample by the spherical K-means.

Figure 15 shows the result of the clustering: with each epoch, the ResNet50-MLP architecture adjusts itself according to the pseudo-labels of the K-Means such that similar samples get clustered. There are



some samples that seem to be misassigned, which may be a result of the t-SNE dimensionality reduction.

Once the training is finished we use the resulting pseudo-labels as LSWR assignments for statistical analysis. Figure 16 shows the mean duration and standard deviation of each LSWR. All LSWRs found have a mean duration between 1–2 days. Some LSWRs, though, show large standard deviations.

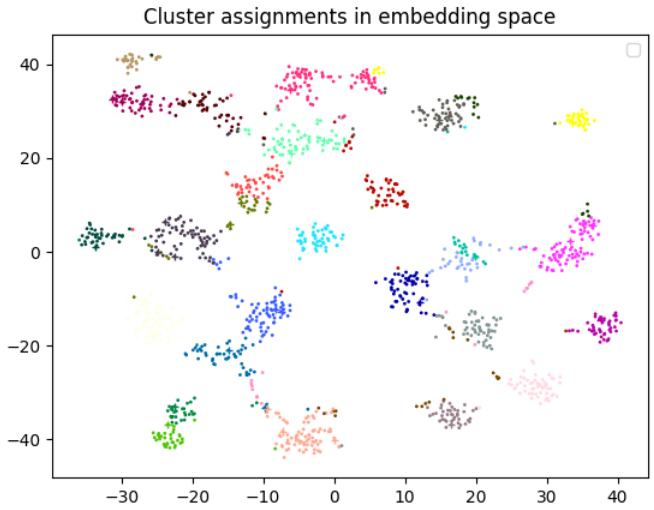


Fig. 15: Result of the spherical K-Means clustering in 2-D embedding space. The output of the MLP has 128 dimensions and is reduced to 2 dimensions using t-SNE.

Hence, the clustering result does not seem very robust yet. We would expect to ideally achieve LSWRs with mean durations up to several days and a lower standard deviation.

As a consequence, we will re-run the experiments with a larger amount of data in terms of time range using the ERA5 dataset. Moreover, we will include more physical quantities by adopting the input layer of the ResNet50 model to allow an arbitrary number of input channels.

Additionally, we want to study the abundance of each LSWR in detail and compare them to the objective and subjective LSWRs of the German Weather Service DWD (*Großwetterlagen*). Furthermore, we aim to create artificial power forecasts using machine learning models and correlate their uncertainties with the LSWRs to see whether they help to assess forecast uncertainty in advance. We will also test whether the LSWRs might help reduce forecast uncertainty in the first place.

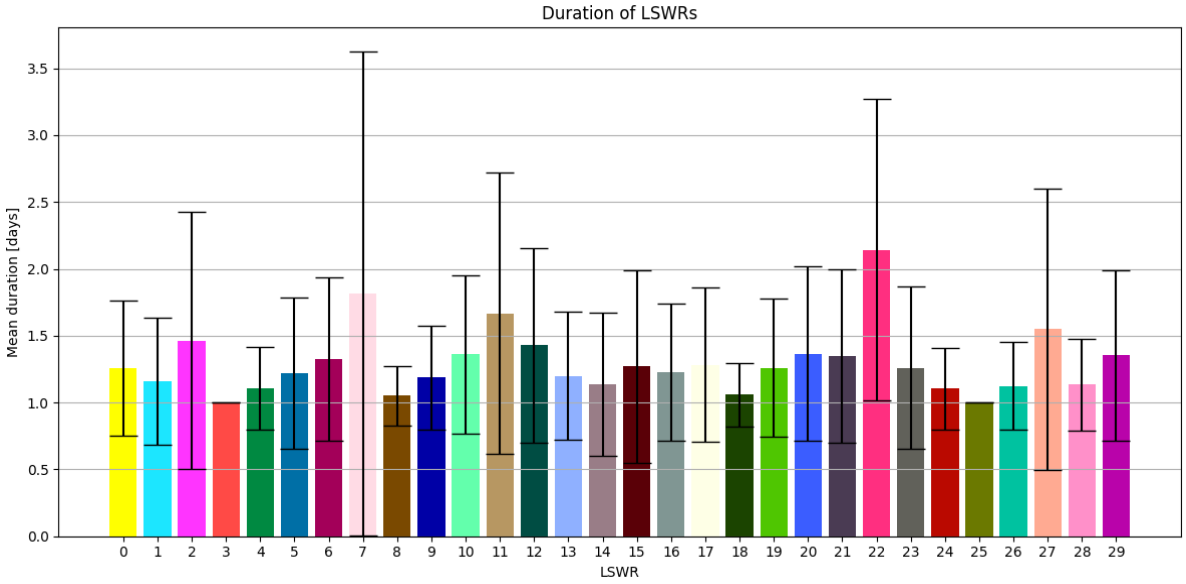
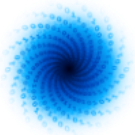


Fig. 16: Mean duration and standard deviation of each large-scale weather regime. The regimes result from the clustering algorithm (cf. Fig. 15) assigning each data sample (day) to a cluster. All regimes have a mean duration of 1–2 days, where some regimes show very large standard deviations (e.g. regime 7).



4 Analysing and improving ML solutions with the Workflow platform Mantik

The Mantik platform aims to assist application developers by streamlining access to external compute sources like the Juelich Supercomputing Center. The user should be able to develop locally and then also execute their application directly from their local machine without the need of connecting to the HPC-system and configuring cumbersome execution scripts. Collaborating on the same model becomes exceedingly difficult as the workflow on HPC-systems depends on the individual users and a lot of access right management would be required to make it work effectively. In addition, the task of monitoring and analysing results is crucial to any ML development. Organising and analysing results that are usually saved in logs or artefacts can be quite cumbersome. This is especially true for HPC clusters, which are only navigable via shell commands and where access and therefore storage is usually only temporarily available to users. Effectively sharing results found in these logs with other users is time consuming as it requires a lot of post-processing to bring the data into a state such that it can be understood and accessed by external users.

Mantik aims to improve the workflow in the previously mentioned areas including HPC access, collaboration, result monitoring and sharing. Since D2.3, Mantik already allows the user to submit their runs to JSC from their local machine via the Mantik CLI. Furthermore, the integration of MLflow into Mantik allows tracking of metrics, parameters and additional artefacts (c.f Fig. 17-19) in real time. This for example enables users to monitor metrics during training and react more quickly to it by cancelling the run or starting promising model variations. In addition, our cloud hosted MLflow GUI allows the user to organise and visualise your results in a user-friendly interface easily accessible by collaborateurs. Since then, the Mantik platform has been further developed and now includes a GUI that forms the main interface to the Mantik platform. Here, we demonstrate how the Mantik platform GUI improves the workflow of ML application developers. For technical details on the Mantik platform we refer to D2.4. In the following, we showcase how Mantik allows the user to submit a run to HPC, tune hyperparameters of the model and analyse their results.

The Mantik GUI introduces *Projects* as the main container that holds all entities relevant for the MLflow life cycle like *Code, Experiments* and *Runs*. *Runs* correspond to any job execution like training or evaluation on the HPC-system. *Code* holds the reference to the git repository that contains the scripts used in *Runs*. *Experiments* are used to organise runs and their tracked metrics. A *Project* can have multiple Experiments to, for example, distinguish between different model types that were tested during project development. This entity based organisation makes reproduction of results very simple, which is always of concern for scientists and usually quite cumbersome.

After setting up *Code* and *Experiment,* we now want to initialise a *Run* to train our model. To do so, we create two configuration files. As previously, Mantik uses an MLflow project file to send the execution command to HPC. In addition, a compute backend file is used to configure the cluster specifications like execution environment and allocated cluster resources. The training *Run* is

created via the form shown in Figure 17. As the parameters specified in the MLproject file can be overwritten in the form and the compute backend configuration freely edited in the GUI, application developers have large flexibility updating their parameters or compute resources. As everything is configured from the platform, tuning hyperparameters for example can now even be done from any device with an internet connection like a mobile phone.

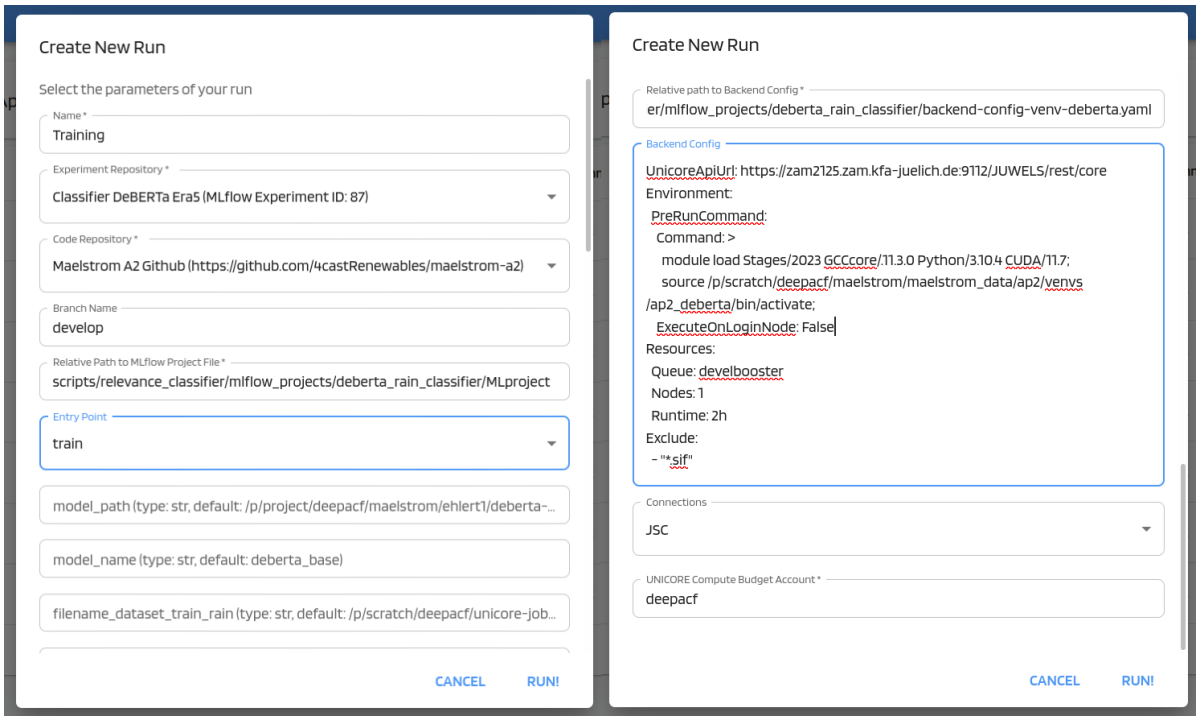


Fig. 17: New runs are configured via this form on the Mantik platform GUI. Both model parameters and cluster specifications are editable from this form, which ensures flexibility for application developers (Note, the same scrollable form is split into two images for clarity).

The MLflow GUI integrated into Mantik allows application developers to track metrics while the model is training. In Figure 18, we provide an example of tracked loss curves from various runs that are continuously updated for models that are still training. Tracking parameters, metrics, figures, etc. only requires adding a single function to the user's script and passing the respective object as an argument. Connecting to the Mantik DB and the hosted MLflow GUI is all handled by Mantik. Therefore, monitoring of metrics becomes easily accessible to Mantik users. The user can, for example, quickly react to unintended model behaviour based on monitored metrics or logged figures.

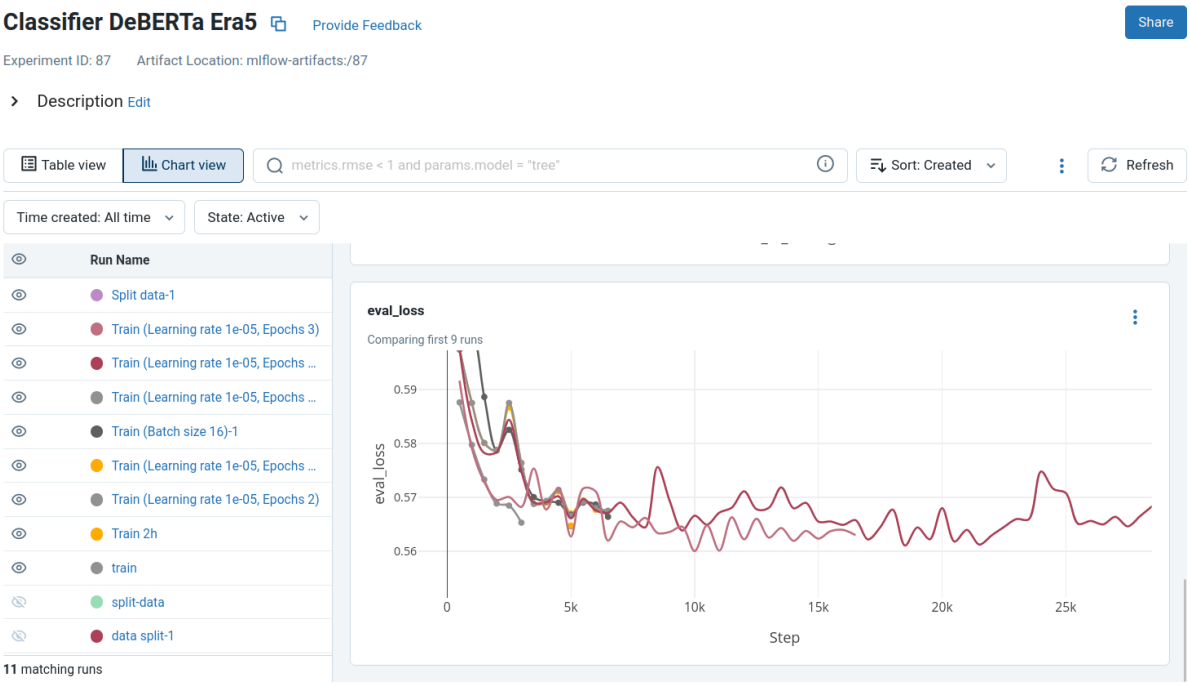


Fig. 18: Tracking performed for our Runs can be visualised via the integrated MLflow GUI, which allows the user to create figures for a quick analysis of their results.

After setting up multiple model variants that vary hyperparameters such as batch size and learning rate, we can analyse our results on the Mantik platform. The MLflow GUI provides functionality to analyse the relationship between tracked metrics and parameters. For this, the user can choose from a variety of plot types like bar, line, scatter and contours charts. In Figure 19, we provide an example of a parallel chart that links the impact of varying batch size, learning rate and epochs to the model performance. The classification problem with tweets as input to infer information on the weather state in AP2 serves here as an example. The model performance is thereby evaluated in terms of the f1-score computed on the evaluation set “eval_f1_raining” and “eval_f1_not_raining” for the classes “raining” and “not raining”, respectively.

Similarly, results from newly introduced model classes can be compared to these baseline metrics in the future. From the GUI, only a few clicks enable these comparisons, which otherwise would require serious manual labour. This workflow is also completely independent from the remote machine used. Therefore, losing access to the machine or continuing development on a different machine would not affect these analyses. Sharing results becomes as easy as inviting collaborateurs or supervisors to the respective Mantik *Project*, which likely heavily reduces preparation time for such meetings. Others can even quickly make their own analysis or monitor performance on their own time.

We plan to further develop the platform by introducing deployment capabilities desired by the applications and provide an even smoother user experience based on feedback in the co-design-cycle.

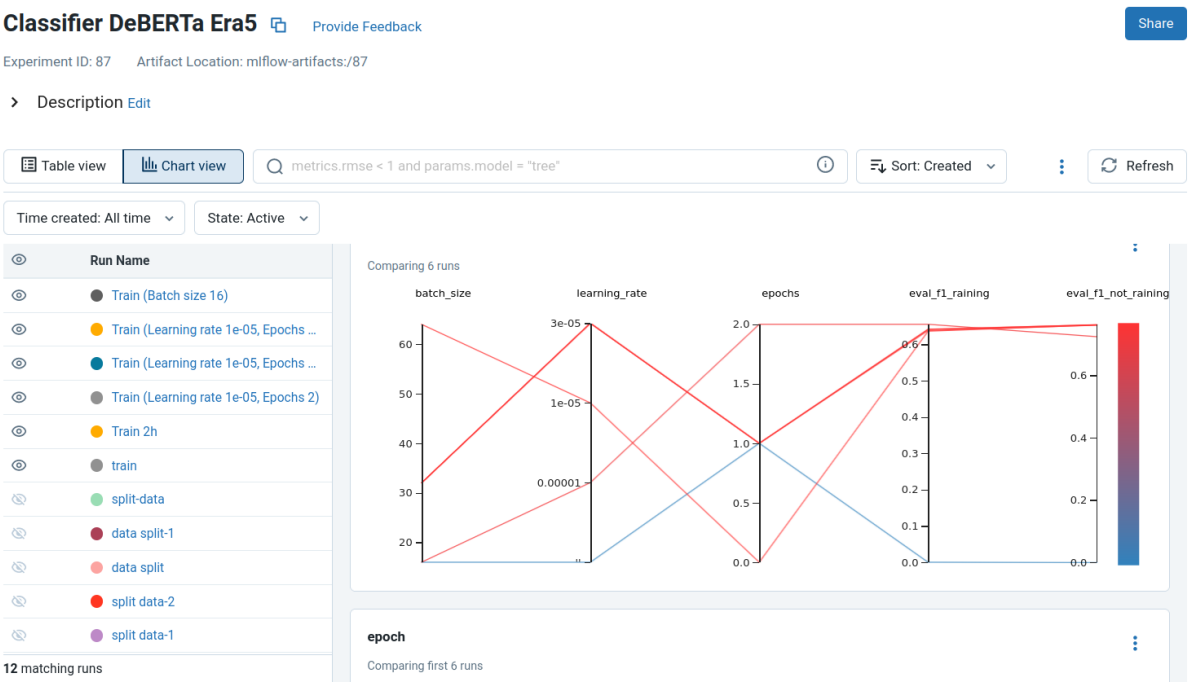
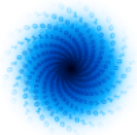


Fig. 19: Tracking performed for our Runs can be visualised via the integrated MLflow GUI, which allows the user to create figures for a quick analysis of their results.



5 References

Ashkboos, Saleh, et al. "ENS-10: A Dataset For Post-Processing Ensemble Weather Forecasts." Advances in Neural Information Processing Systems 35 (2022): 21974-21987.

Ben-Bouallegue, Zied, et al. "The rise of data-driven weather forecasting." arXiv preprint arXiv:2307.10128 (2023).

Bi, K., L. Xie, et al. “Accurate medium-range global weather forecasting with 3D neural networks”. Nature, doi:https://doi.org/10.1038/s41586-023-06185-3. (2023).

Bommasani, Rishi, et al. "On the opportunities and risks of foundation models." *arXiv preprint arXiv:2108.07258* (2021).

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A., “Unsupervised Learning of Visual Features by Contrasting Cluster Assignments”, *arXiv e-prints*, *doi:10.48550/arXiv.2006.09882* (2020).

OpenAI “GPT-4 Technical Report” *arXiv preprint arXiv:2303.08774 (2023).*

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929* (2020).

Hasselmann, Klaus. "Stochastic climate models part I. Theory." *Tellus* 28.6 (1976): 473-485.

Kowsari, K., Meimandi, K. J., Heidarysafa, M., Mendu, S., Barnes, L. E., & Brown, D. E. “Text Classification Algorithms: A Survey.” Information, 10 (4), 150 (2019).

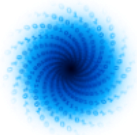
Lagerquist, R., Turner, D., Ebert-Uphoff, I., Stewart, J. and Hagerty, V., “Using deep learning to emulate and accelerate a radiative transfer model. Journal of Atmospheric and Oceanic Technology”, 38(10), pp.1673-1696 (2021).

Lessig, Christian, et al. "AtmoRep: A stochastic model of atmosphere dynamics using large scale representation learning." arXiv preprint arXiv:2308.13280 (2023).

Met Office. ”Met Office MIDAS Open: UK Land Surface Stations Data (1853-current)”. Centre for Environmental Data Analysis. (2019).

Muñoz Sabater, J. "ERA5-Land hourly data from 1950 to present. Copernicus Climate Change Service (C3S) Climate Data Store (CDS)". DOI: 10.24381/cds.e2161bac (Accessed on 27-04-2023) (2019)

Palmer, T. N., and Paul David Williams. "Introduction. Stochastic physics and climate modelling." *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 366.1875 (2008): 2419-2425.*



Rasp, Stephan, et al., 2023. "WeatherBench 2: A benchmark for the next generation of data-driven global weather models." arXiv preprint arXiv:2308.15560.

Schmid,Philipp, et al. "Spread Your Wings: Falcon 180B is here" *Web blog post. https://huggingface.co/blog, Hugging Face,* (2023).

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

Yao, Y., Zhong, X., Zheng, Y. and Wang, Z. "A Physics-Incorporated Deep Learning Framework for Parameterization of Atmospheric Radiative Transfer". Journal of Advances in Modeling Earth Systems, 15(5), p.e2022MS003445 (2023).

You, Y., Gitman, I., and Ginsburg, B., "Large Batch Training of Convolutional Networks", *arXiv e-prints*, *doi:10.48550/arXiv.1708.03888* (2017).

Zhou, Ce, et al., "A comprehensive survey on pretrained foundation models: A history from Bert to ChatGPT". *arXiv preprint arXiv:2302.09419* (2023).

Document History

Version	Author(s)	Date	Changes
V1	Michael Langguth et al.	27/09/2023	Version for review
V2	Michael Langguth et al.	30/09/2023	Final Version

Internal Review History

Internal Reviewers	Date	Comments
Peter Dueben (ECMWF)	27/09/2023	Minor comments and suggestions added

Estimated Effort Contribution per Partner

Partner	Effort
ECMWF	0.5 PMs
MetNor	
4cast	
FZJ	2 PMs
Total	**2.5**