DSM500 Final Project Report[1]

Goldsmiths
UNIVERSITY OF LONDON

University of London, International Programmes
MSc Data Science

Enhancing GraphCast for Improved Weather Forecasting

Stefano Schuppli (SRN 150194831)
March 2025

Institution: Self-study
Supervisors: Luca Parisi

[1]Submitted as part of the requirements for the award of the Degree in Data Science of the University of London.

Abstract

This study investigates NVIDIA re-implementation of GraphCast, a Graph Neural Network model for data-driven medium-range global weather forecasting originally published by Google DeepMind. NVIDIA re-implemented GraphCast in PyTorch within its Modulus framework but did not release pre-trained weights, raising concerns about the ability of their re-implementation to achieve the expected predictive performance. Operated with limited computational resources, this project assesses whether meaningful predictive performance can be achieved through this codebase and proposes a series of enhancements to it. A systematic benchmarking framework is introduced to assess the abilities across different lead times, meteorological variables, and geographic regions. The proposed enhancements are shown to substantially increase its weather forecasting skills. This report is intended for research groups considering the adoption of this GraphCast implementation, and it thus aims to contribute to the field of data-driven weather forecasting.

Keywords: Machine Learning for Weather Forecasting, Graph Neural Networks, GraphCast, NVIDIA Modulus.

Word Count

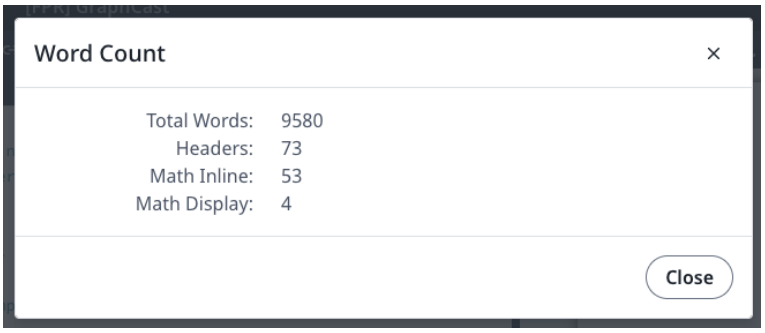


Figure 1: Word count obtained from Overleaf after excluding permissible contents (e.g. appendices).

Contents

1 Introduction **4**
1.1 Project Aims . . . 4
1.2 Research Questions . . . 5
1.3 Professional Context . . . 5
1.3.1 Infrastructure Constraints . . . 6
1.4 Statement of Relevance . . . 6
1.5 Ethical and Legal Considerations . . . 6
1.6 Statement on the Usage of LLM-based Services . . . 7

2 Background **8**
2.1 Data-Driven Weather Forecasting . . . 8
2.2 Models for data-driven weather forecasting . . . 9
2.3 Graph Neural Networks (GNNs) . . . 10
2.4 Datasets . . . 11
2.4.1 ERA5 . . . 11
2.5 Common Variables of Interest . . . 12
2.6 Common Evaluation Metrics . . . 12
2.7 Common Methods for Models Comparison . . . 14

3 Methods **15**
3.1 Dataset . . . 15
3.2 Codebase Adoption and Model Architecture . . . 16
3.2.1 Learning Rate Schedule and Training Program . . . 17
3.2.2 Loss Function and Optimizer . . . 18
3.2.3 Reproducibility Instructions . . . 18
3.3 Results Evaluation Protocol . . . 19
3.3.1 Metrics Reported . . . 19
3.3.2 Results Collection . . . 20
3.3.3 Baselines and Performances Recordings . . . 20
3.3.4 Results Presentation and Visualization . . . 20
3.3.5 Statistical Comparison . . . 20
3.4 Experimental Approach . . . 21
3.5 Infrastructure . . . 22

4 Results **23**
4.1 Modulus' GraphCast Dry-Run . . . 23
4.2 Inference Framework . . . 25

4.3 Code Base Analysis and Refactoring . . . 28
4.4 Performance Baseline . . . 29
4.5 Optimal Learning Schedule . . . 30
4.6 GraphCast Enhancements . . . 32
4.6.1 Sea-Surface Temperature Effect on Inverse Variance Weights . . . 32
4.6.2 Improved Smoothness for Temporal Information . . . 33
4.6.3 Misalignment of Top-of-Atmosphere Solar Irradiance . . . 35
4.6.4 Seasonal Bias Removal . . . 35
4.7 Ablation Study . . . 36
4.7.1 Sea-Surface Temperature Channel . . . 37
4.7.2 Temporal Information Smoothness Enhancement . . . 38
4.7.3 Misalignment of Top-of-Atmosphere Solar Irradiance . . . 38
4.7.4 Seasonal Bias Removal . . . 38
4.8 FourCastNet as an External Reference . . . 39
4.9 Further Experimentation . . . 39
4.9.1 Model Size Scaling . . . 39
4.9.2 Phase 3 Learning Rate Levels . . . 40
4.10 Improved Model Performance . . . 42
4.11 Out-of-Sample Model Performance Results . . . 43
4.11.1 Quantitative Performance Results . . . 45
4.11.2 Inference Artifacts Samples . . . 47
4.12 Statistical Significance Tests . . . 52

5 Discussion **53**

6 Conclusions and Future Directions **54**

References **57**

Appendices **61**

A Abbreviations and definitions **62**
A.1 Abbreviations . . . 62
A.2 Definitions . . . 62
A.2.1 Data Assimilation . . . 62
A.2.2 Reanalysis . . . 63
A.2.3 Variables and Channels . . . 63
A.2.4 Surface and Pressure Level Variables . . . 63
A.2.5 Probabilistic Forecasting . . . 64

B A primer on GNNs **65**

C Other Weather Datasets **67**

D Extended Out-of-Sample Model Performance Results **69**

1. Introduction

The field of weather forecasting is undergoing a major transformation, with traditional physics-based numerical methods increasingly challenged by data-driven approaches such as Machine Learning (ML). By leveraging vast amounts of observational and simulated weather data, these novel approaches produce forecasts more efficiently and with accuracy levels comparable to traditional methods. Though their operational use is still limited[1], they represent a compelling alternative for future forecasting systems. As a result, the research field of ML applied to weather forecasting is experiencing considerable activity.

The Background chapter presents an overview of recently published models. They share the objective of generating accurate and reliable medium-range forecasts (i.e., 3 to 15 days ahead) on a global scale. Among these models, GraphCast, published by Google DeepMind, is based on a Graph Neural Network (GNN) architecture. As a research code it is not actively maintained. Its JAX-based implementation represents another element of friction for its adoption.

NVIDIA re-implemented GraphCast in PyTorch as part of their Modulus product, making the GraphCast architecture accessible to teams interested but constrained to using PyTorch, e.g., due to available competences or to the PyTorch extensive ecosystem.

While NVIDIA has released the re-implementation code (as part of the Modulus suite[2]), it has not done so for the weights. This contrasts to what NVIDIA has done for other models of the Modulus suite, whose weights are available for download from their NGC Catalog[3].

This omission raises questions regarding the NVIDIA re-implementation's ability to reproduce the performance of the original DeepMind publication. Consequently, this uncertainty makes it difficult to assess its suitability for adoption, especially given the non-negligible costs of a full-training campaign.

1.1 Project Aims

This project aims to reduce the stated lack of clarity surrounding NVIDIA's re-implementation of GraphCast by evaluating its ability to develop predictive power.

If NVIDIA's version fails to achieve accuracy levels equivalent to the DeepMind publication, it could be due to gaps in their implementation or training protocol. Through a focused investigation of its architectural aspects, training dynamics, and performance characteristics,

[1] `https://www.ecmwf.int/en/about/media-centre/news/2025/ecmwfs-ai-forecasts-become-operational`

[2] `https://github.com/NVIDIA/modulus`, renamed from `Modulus` to `PhysicsNeMo` on March 18th, 2025.

[3] `https://catalog.ngc.nvidia.com/models`

this work seeks to determine whether meaningful forecasting capabilities can be achieved through this implementation.

In addition, it aims to provide insight into the expected performance trajectory if scaled to a full training campaign, helping assess its viability for broader adoption in research and operational contexts.

1.2 Research Questions

Under constrained resources, this project evaluates the ability of NVIDIA's GraphCast re-implementation to produce predictive power. This effort is guided by the following questions.

1. **What level of forecasting capability is achievable through this re-implementation?**
2. **How does the predictive performance of this model compare to a reference baseline?**
3. **What effect do architectural and hyperparameters choices have on the model accuracy, stability and generalization?**

The process of answering such questions will require analysing the codebase, which should provide opportunities for improvement. Thus, the connection to the project title.

1.3 Professional Context

I am part of the Machine Learning (ML) engineering staff at the Swiss National Supercomputing Centre[4] (CSCS), where we are finalizing the installation of Alps[5], a new Research Infrastructure (RI) featuring NVIDIA GH-200 Grace-Hopper[6] GPUs. The users we support are predominantly research groups from Swiss institutions, active in various scientific domains.

Our tasks include testing the infrastructure to better understand its complexities and enhance our ability to support users. Furthermore, while we don't conduct ML research ourselves, a deep understanding of our users' models and use cases is essential for maximizing our impact.

An area of particular interest to my group stands at the intersection between ML and Weather and Climate forecasting. This area is experiencing a rapid evolution, with new breakthrough approaches being published at a relatively fast pace (see Background).

The PyTorch re-implementation of GraphCast[7], found in Modulus, is currently attractive to our users. It allows PyTorch-experienced researchers to quickly explore ideas avoiding the JAX learning curve, and benefits from Modulus's open-source nature and NVIDIA's support, prespecting long-term continuity that is absent in many research codes.

By adopting such a model as the focus of this project, the effort involved in this work aligns with the interests of my employer. This choice makes it possible, up to a certain extent (see sub-section), to use our exceptional infrastructure to carry out this project.

[4] `https://www.cscs.ch`

[5] `https://www.cscs.ch/computers/alps`

[6] `https://www.nvidia.com/en-us/data-center/grace-hopper-superchip`

[7] `https://docs.nvidia.com/deeplearning/modulus/modulus-core/examples/weather/graphcast/readme.html`

To ensure integrity, the work performed for this project must remain distinct from ongoing work activities, especially group efforts, at least until after its marking.

1.3.1 Infrastructure Constraints

The following constraints must be met in order to use the Alps RI for this project.

1. The outcome should be relevant to my employer.
2. No dedicated compute or storage budget is allocated. This project must be carried out using idle nodes, e.g., from the debug queue[8], and public datasets already present on the storage systems.
3. Activities of our end-users must not be hindered. Care must be taken to avoid negative effects on shared system resources, such as network or storage bandwidth.

Consequently, it is not possible to train GraphCast to convergence (w.r.t., full-training schedule).

1.4 Statement of Relevance

This MSc project does not aim to develop a fully trained operational model for meteorological agencies, as this exceeds the available computational budget and domain knowledge.

Instead, it evaluates whether NVIDIA's GraphCast re-implementation can achieve meaningful predictive performance and how modifications impact its accuracy. The findings will support research teams interested in adopting this codebase, enhance CSCS's ability to assist users and join collaborations, thus aligning with my employer's interests.

Ultimately, this work may indirectly benefit sectors that depend on weather forecasting, such as agriculture, transportation, and disaster planning.

1.5 Ethical and Legal Considerations

The dataset utilized consists of global meteorological data. It does not contain personally identifiable or sensitive information. It is derived from ERA5, a publicly available product of the Copernicus Programme subject to the *Licence to Use Copernicus Products*[9]. Bias and regional disparities (e.g. due data source density) are discussed in the Background chapter.

The codebases utilized are obtained through the respective Github repositories:

- NVIDIA Modulus[10], the focus of this project, released under *Apache License 2.0*.
- FourCastNet[11], used for results comparison, released under the *BSD 3-Clause License*.

Research activities are allowed on the above by the respective licenses.

[8] Debug nodes are intended for short development-like workloads, making them unsuitable for large-scale experiments.

[9] `https://apps.ecmwf.int/datasets/licences/copernicus`

[10] `https://github.com/NVIDIA/modulus`, renamed from `Modulus` to `PhysicsNeMo` on March 18th, 2025.

[11] `https://github.com/NVlabs/FourCastNet`

Due to infrastructural constraints, it will not be possible to produce weights for the operationalization of this model. The risk of misguiding decision making, e.g. in life-threatening contexts, is deemed marginal.

The repository `https://github.com/schuups/DSM500` contains what is needed for results reproducibility. The `GoldSmithsMScDataScience` GitHub account has been granted access.

1.6 Statement on the Usage of LLM-based Services

Services such as Writefull[12] and ChatGPT[13], and Gemini[14] have been used in the production of this report to make texts more concise, identify redundant information, and improve readability. All generated output was manually reviewed to ensure accuracy and consistency with original thoughts. None of the ideas, concepts, or conclusions expressed in this report originated from these services.

It can be assumed that this quality-oriented approach has been applied to each paragraph of this report and that the most common prompt used was "*make this text more concise*". As such, it can be viewed as an evolution of traditional text-editing tools such as spell checkers and thesaurus features.

This approach received positive feedback from the project supervisors[15].

[12] `https://www.writefull.com/writefull-for-overleaf`
[13] `https://chat.openai.com`
[14] `https://gemini.google.com`
[15] `https://learn.london.ac.uk/mod/forum/discuss.php?d=66673#p164835`

2. Background

2.1 Data-Driven Weather Forecasting

Improving weather forecast accuracy requires higher spatial and temporal resolutions to model increasingly localized phenomena such as cloud microphysics [1]. However, traditional physics-based models face significant, possibly insurmountable, computational challenges in achieving these resolutions [2], [3]. Paraphrasing [1], each doubling of horizontal resolution increases computational requirements between 8-16 times.

To address such scaling challenges, several approaches have been proposed. Hybrid modelling techniques integrate physics-based models with statistical post-processing to improve forecast efficiency while maintaining physical consistency [3]. To simplify complex physical equations, other approaches focus on dominant dynamics caring to minimise the loss of predictive skill [1], [4]. Data-driven approaches leverage historical and observational data to produce rapid and cost-effective predictions [5], [6].

The availability of high-quality datasets [7] and hardware suitable for deep neural networks (DNNs) has enabled rapid advances in ML also in the field of weather forecasting [8]–[11]. These advances were further supported by the introduction of specialized tools, such as WeatherBench2 [12] for standardized evaluation protocols.

The level of skill already achieved by data-driven approaches is comparable to traditional NWP approaches and, in some cases, exceeds it [13]. Moreover, this novel type of forecasting requires a fraction of the computational cost.

Meteorological agencies are investing in both the development of models and in the integration of operational workflows. The Artificial Intelligence Forecasting System (AIFS) [14], developed by the European Centre for Medium-Range Weather Forecasts (ECMWF), is an example which recently[13] became operational and is now being operated alongside traditional systems.

Their adoption is still hindered by remaining challenges. The lack of interpretability (physical consistency?) limits their applicability in high-stakes scenarios [3], [4]. Their reliance on historical data raises reliability concerns, e.g., about extreme events or mutated climate conditions [15], [16]. Further research into hybrid approaches, uncertainty quantification, and benchmarking is needed to support acceptance in operational settings [6], [17], [18].

2.2 Models for data-driven weather forecasting

Models for weather forecasting typically process data in the form of tridimensional grids: stacked images of the globe representing different meteorological variables, such as temperature, wind speed, and solar exposure. Each time step has an associated 3D grid making the dataset four-dimensional. Different DL techniques (e.g. CNNs, transformers, graphs) are applied to learn spatio-temporal dependencies.

Autoregression is commonly used to produce prediction rollouts, predicting future states (after the 1st step) from previous outputs.

Several prominent models have emerged in just a few years. This section provides a brief overview.

- **FourCastNet** [17], [19], [20] (2022-2024): This model applies Adaptive Fourier Neural Operators (AFNOs) and Spherical FNOs (SFNOs) to capture atmospheric dynamics in the spectral domain. The spectral domain represents weather patterns as combinations of frequencies, thus enabling the modelling of atmospheric variations at different spatio-temporal scales.
- **Pangu-Weather** [21] (2022): This model employs a 3D transformer architecture to model spatio-temporal dependencies at multiple scales. The atmosphere is divided in cubes, processed independently through self-attention mechanisms to maintain long-range dependencies.
- **GraphCast** [22] (2023): GraphCast employs GNNs to model spatial dependencies at different scales. The gridded input is mapped into an icosahedral-shaped graph representing Earth. The multi-mesh structure overlays multiple resolution levels of the Earth's surface allowing for the processing of information across different spatial scales. Similar to Pangu-Weather, it was trained to generate 6-hour forecasts up to 10 days in advance at a 0.25° resolution.

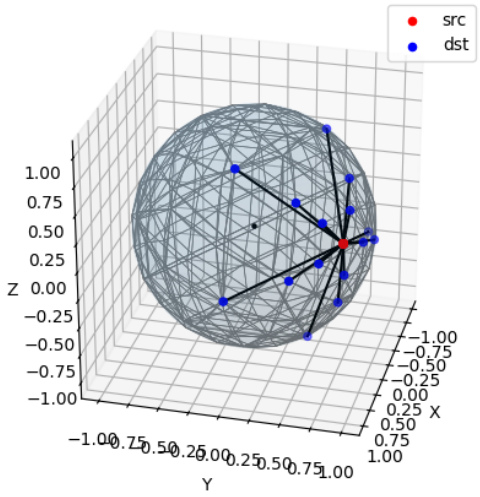


Figure 2.1: Multi-scale node connections in the graph structure employed by GraphCast.

- **FuXi** [23], [24] (2023-2024): FuXi uses a cascaded approach for short-, medium-, and long-term forecasts, enhancing accuracy on longer time horizons. Short-term predictions

feed into longer-term forecasts, progressively refining outcomes. A hierarchical training strategy optimizes focus based on lead time, balancing between accuracy and efficiency. Ensemble strategies combine multiple runs' outputs for uncertainty estimation.

- **GenCast** [18] (2024)[1]: GenCast uses generative diffusion models to create probabilistic forecasts by refining random noise into realistic weather states through denoising steps. This generates diverse weather scenarios for risk assessment and extreme event prediction, introducing stochasticity at each step. Unlike deterministic models, GenCast captures forecast uncertainty without needing explicit perturbations or ensemble methods.
- **GraphDOP** [15] (2024)[1]: GraphDOP processes raw observational data, like satellite images and weather readings, reducing reliance on NWP reanalysis. Its architecture adapts to data density, enhancing real-time forecasting flexibility with graph-based techniques for modeling spatio-temporal dependencies.

2.3 Graph Neural Networks (GNNs)

The Appendix includes a primer on GNNs.

GNNs operate on graph-structured data, capturing relational dependencies among entities. They find application in domains such as social network analysis [25], molecular chemistry [26], and physics simulations [27]. While strong in learning dependencies, GNNs can suffer from over-smoothing (where node representations become too similar [28]) and sensitivity to connectivity (where poor connectivity may impede message passing [29]).

GNNs are increasingly applied to physical simulations such as weather forecasting. A review of key publications follows.

- **Mesh-Based and Multi-Scale MeshGraphNets** [27], [30] (2021, 2022): This work uses GNNs to simulate physical systems with adaptive mesh structures, varying (before training) the density of the nodes over the simulation space. Allocating higher resolution where finer details are required optimizes computational resource.
- **Forecasting Global Weather with GNNs** [9] (2022): Introduces a GNN-based model for forecasting global weather, predicting atmospheric states every six hours. It achieves performance comparable to traditional numerical methods at 1-degree resolution, showing the potential of GNNs in medium-range forecasting.
- **GraphCast** (the subject of this report) [22] (2023): GraphCast improves [9] by using a multi-mesh structure to model also longer-range interactions, a deeper GNN processor with unshared weights, demonstrating improved accuracy. It includes a wider evaluation of severe weather, increasing real-world applicability.
- **AIFS** [14] (2024): A data-driven weather forecasting system that uses (at least in its earlier versions) a GNN-based encoder-decoder and a sliding window transformer. It was promoted to operational in February 2025 [13].
- **AIFS-CRPS** [31] (2024): AIFS-CRPS enhances AIFS with a probabilistic method, substituting MSE loss with afCRPS (Continuous Ranked Probability Score) to improve

[1]These works were published during the execution of this MSc project.

probabilistic calibration and minimise ensemble size bias. It surpasses ECMWF's IFS ensemble in medium-range forecasts and is competitive in sub-seasonal predictions.

2.4 Datasets

This section describes the dataset used for training and evaluating GraphCast in its original study [22, Section 1.1] and contrasts it with the subset [32] selected for this project (w.r.t., Infrastrucure Constraints). The evaluation of alternative datasets, detailed both here and under Other Weather Datasets Appendix, identified ERA5 as the *gold standard* for research in this field, and it therefore supports its adoption as a suitable choice for this project.

2.4.1 ERA5

ERA5 [7] is a dataset generated through the reanalysis of historical observational data[1].

ML research in weather forecasting predominantly employs the ERA5 dataset due to its high-quality, spatial and temporal resolution, historical range, and accessibility. All reviewed models rely on ERA5, with the exception of GraphDOP which processes observational data directly.

Produced by the Copernicus Climate Change Service (C3S) at ECMWF, ERA5 is the fifth-generation reanalysis of global weather, providing the best estimates of what the global atmospheric state was, on an hourly basis, from 1940 to today.

The dataset includes variables such as temperature, wind speed, and humidity. These are available across different elevations, from the surface to the uppermost regions of the atmosphere, with the vertical position represented by the air pressure (in hPa). It features a $0.25° \times 0.25°$ horizontal resolution on a 721×1440 (latitude-longitude) grid, with points at the equator representing an area of approximately 31 km^2.

While the dataset can be queried through the Copernicus' Climate Data Store[2] (CDS), extensive slices are impractical. Research often employs subsets, like [33] using 137 variables (about 500 TB) and [32] using 21 (about 5 TB).

Due to sparse observations, data quality before the satellite era is lower, so ML research generally uses data from 1979. Quality may also be affected by the Northern Hemisphere's higher density of observational data sources, including weather stations, aircraft, and ships. Section 2.7 outlines typically adopted methods to address this bias. Extreme weather event data and, to some extent, precipitation are under-represented, requiring specialized datasets for model evaluation. Appendix C reviews one such dataset.

Section 3.1 of [12] summarizes other key caveats of ERA5: (1) being a model simulation, some data, like precipitation, may deviate from actual observations [34], and (2) the assimilation window influences data quality [22]. The ERA5 documentation[3] discusses the topic of "Accuracy and uncertainty" in greater detail.

[1] Reanalysis datasets are generated by running NWP simulations on sparse observational data (e.g., weather station measurements), creating a dense and consistent record of Earth's atmospheric conditions at different points in time. See also Appendix A

[2] `https://cds.climate.copernicus.eu`

[3] `https://confluence.ecmwf.int/x/wv2NB`

2.5 Common Variables of Interest

Assessment of model abilities to learn weather dynamics typically focuses on several key variables, also referred to as *headline variables*. Extrapolating from [12], [17]–[24], these are:

- **500 hPa Geopotential (z500)** is located in the mid-troposphere[4] (approx. 5-6 km altitude a.s.l.), where significant weather patterns like ridges[5] and troughs[6] are clearer. These correlate well with broader lower-level weather patterns, such as surface temperature. A model predicting this level accurately likely performs well on others.
- **850 hPa Temperature (t850)** at about 1.5 km a.s.l. is more stable against diurnal heating and terrain shape, making it a reliable indicator of air mass traits such as advections[7], frontal boundaries[8], and cyclone development[9]. It is commonly used in weather forecasting due to its strong link to near-surface temperature patterns.
- **2-meter Temperature (t2m)** is essential for predicting human-relevant conditions like heatwaves and cold spells. However, it's influenced by terrain, urban heat islands, and daily variations, making it more volatile than t850.
- **Mean Sea Level Pressure (msl)** reveals high- and low-pressure systems, crucial for identify weather patterns gradients which drive wind and influence storms.
- **10-meters and 850 hPa Wind Speeds (10u, 10v, u850, v850)** are crucial for energy applications and severe weather detection. Winds at 850 hPa are key for identifying factors affecting cyclones.

Precipitation is not included in the dataset but is a common variable of interest.

2.6 Common Evaluation Metrics

ML research on weather forecasting has adopted, in addition to standard DL evaluation methods, also established domain-specific ones. See also [35, at around 13:28].

The brief overview that follows is extrapolated from the works of [12], [17]–[24]. It does not include metrics that are oriented towards probabilistic forecasts. This is primarily due to GraphCast being a deterministic model.

Latitude-Weighted Errors (MSE and RMSE)

Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) [36] are widely used for quantifying the magnitude of forecast errors. These are typically latitude-weighted to address poles data over-representation in the grid-shaped data.

[4]The troposhpere is the lowest region of the atmosphere, extending from the earth's surface to a height of about 6–10 km. The dataset in use includes levels in this region.

[5]A higher pressure area typically associated with warm air masses, and fair weather such as clear skies.

[6]A lower pressure area, typically associated with colder air masses, clouds and precipitations.

[7]Advection is the horizontal movement of air that carries properties like temperature or moisture from one place to another.

[8]A frontal boundary is the dividing line between two air masses with different temperatures, densities, and humidity levels. It is where these air masses clash, triggering weather changes.

[9]A cyclone is a large-scale system of rotating winds around a low-pressure center. It is characterized by inward-spiraling air and is associated with clouds and precipitation.

Definitions. Given a set of N predictions $\hat{y}_i$ and corresponding ground truth values y_i, the MSE and RMSE are defined as:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} w(\hat{y}_i - y_i)^2 \tag{2.1}$$

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} w(\hat{y}_i - y_i)^2} \tag{2.2}$$

where:

- w is usually defined as the cosine of the latitude value ϕ and therefore independent of the sample i. It is supplied in a shape compatible with the shape of y.
- The summation runs over all spatial-grid points in the domain.

Intuition. MSE calculates the average squared difference between predicted and actual values, penalizing larger errors more, and hence it is sensitive to outliers. RMSE shares the unit of the original variable and is easier to interpret. It thus reflects the standard deviation and typical size of prediction errors.

Strengths and Limitations. For weather forecasting applications, MSE and RMSE are considered to have drawbacks: (1) RMSE isn't normalized, complicating comparisons across variables like temperature and pressure. (2) MSE fails to capture structured weather dependencies, leading to high errors for accurate forecasts that are slightly shifted in space or time. (3) MSE's error squaring overemphasizes rare extremes, skewing overall skill assessment and hindering extreme event predictions. (4) MSE doesn't offer insights into physical forecast quality, like atmospheric feature evolution.

Bias Analysis. While MSE and RMSE quantify error magnitudes, bias analysis examines whether a model systematically over- or under-predicts atmospheric variables. Bias is simply the mean difference between forecasts and observations, thus the same as MSE but without the squaring operation. A positive value indicates overestimation and a negative value indicates underestimation. Bias can be assessed globally, regionally, or across different forecast lead times. Bias maps can help identify systematic forecasting errors. While lower bias suggests improved calibration, it should be analyzed alongside RMSE and ACC (see next section) to ensure meaningful improvements rather than simple error shifting.

Latitude-Weighted Anomaly Correlation Coefficient (ACC)

The Anomaly Correlation Coefficient (ACC) [36], [37] evaluates the spatial correlation between forecast anomalies and observed anomalies, assessing forecast accuracy for deviations from the climatological mean and serving as a key metric for verifying large-scale weather patterns.

Definition.

$$\text{ACC} = \frac{\sum_{i=1}^{N} w(\hat{y}_i - \bar{y}_i)(y_i - \bar{y}_i)}{\sqrt{\sum_{i=1}^{N} w(\hat{y}_i - \bar{y}_i)^2}\sqrt{\sum_{i=1}^{N} w(y_i - \bar{y}_i)^2}} \tag{2.3}$$

where:

- $\bar{y}_i$ represents the climatological mean over a reference period.
- $\hat{y}_i - \bar{y}_i$ and $y_i - \bar{y}_i$ are the forecast and observed anomalies, respectively.
- w, as for MSE and RMSE, is the latitude weight.
- The summation runs over all spatial-grid points in the domain.

Intuition. ACC evaluates how accurately a forecast captures anomaly patterns instead of exact values. High ACC occurs when forecasts maintain general trends such as warm areas staying warm and cold areas staying cold. A perfect forecast gives ACC = 1; an uncorrelated one gives ACC = 0. Negative values show an inverse correlation.

Strengths and Limitations. ACC is resilient to systematic biases by prioritizing anomaly structures over absolute values, which is important for longer-term forecasts involving structured patterns. However, ACC can be misleading if it captures general anomalies but misses finer details as it does not measure error magnitude, allowing a forecast with low RMSE to still have low ACC if anomalies are wrongly placed.

2.7 Common Methods for Models Comparison

Comparing the predictive ability of two deterministic models usually relies on metrics like ACC, MSE, and RMSE [12], [17], [22], [38]. Advanced scores like SEEPS or CRPS may not apply to deterministic models, such as GraphCast. Studies often support the interpretation of headline metrics[10] results through visualization (e.g., line plots) and scorecards[11].

While aggregate metrics offer a quick view of the models performance, they can prelude the assessment of more nuanced predictive skills. Therefore, studies often present them not only as global averages, but also in more granular segments, such as by variables, by lead times, or by geographical regions. This helps identify specific strengths and weaknesses, preventing important variations from being overlooked. Figure 1 of [38] and Figure 17 of [22] are examples of such data products.

[10] Headline scores are key metrics (e.g., ACC on geopotential at 500 hPa) elected to faciliate the tracking of model performance improvements year-over-year.

[11] Scorecards provide a grid of metrics across multiple variables, regions, and lead times, highlighting where forecasts excel or underperform.

3. Methods

3.1 Dataset

Due to the infrastructure constraints described in Section 1.3.1, this work uses the same 5 TB ERA5 subset employed by FourCastNet [17], [32].

Compared to the variables used in the original GraphCast publication, this subset lacks vertical air velocity (`w`), uses relative humidity (`r`) in place of specific humidity (`q`), and includes both surface pressure (`sp`) and sea surface temperature (`sst`). Total column water vapour (`tcwv`) is used due to its strong predictive relationship of precipitation, whereas total precipitation (`tp`) data itself is not used. In terms of vertical levels, it covers 2 to 4 levels instead of 37 per atmospheric variable used in [12], [22].

Both have the same temporal sampling (6-hourly) and spatial resolution (global 721×1440 grid). The period covered is shorter: 1979-2018 vs. 1979-2021. The FourCastNet [17] split, being similar to the GraphCast study [22] is adopted: it uses 1979-2015 for training, 2016-2017 for testing, and 2018 for validation (named *out-of-sample*).

Despite the comparatively small size of this subset, FourCastNet [17] has demonstrated that it can be modeled effectively, suggesting its suitability for this project aims.

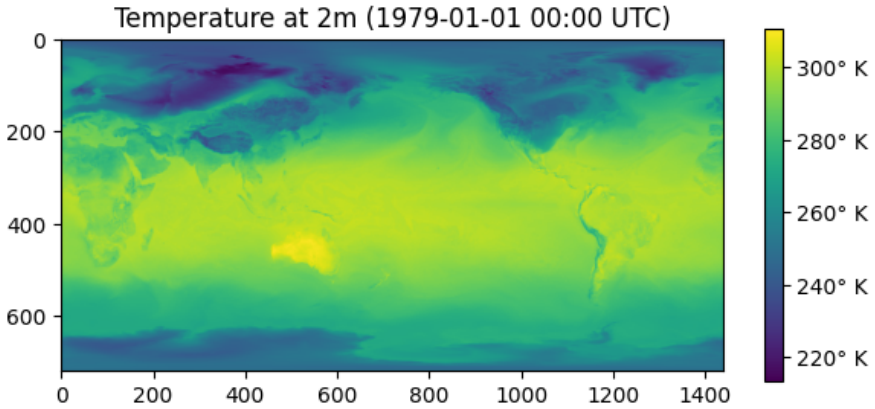


Figure 3.1: The temperature at 2 meters channel from a subset sample.

The model operates on channels loaded from the dataset as well as static and runtime-generated ones. These are detailed in Table 3.1, of which the channel of a sample is visualized in Figure 3.1.

Type	Description	Short name	ECMWF ID	Vertical levels (in hPa)
Surface	Wind eastward component at 10 meters	10u	165	-
Surface	Wind northward component at 10 meters	10v	166	-
Surface	Temperature at 2 meters	t2m	167	-
Surface	Surface pressure	sp	134	-
Surface	Mean seal level pressure	msl	151	-
Surface	Total column water vapour	tcwv	137	-
Surface	Sea surface temperature	sst	34	-
Atmospheric	Temperature	t	130	500, 850
Atmospheric	Relative humidity	r	157	500, 850
Atmospheric	Wind eastward component	u	131	500, 850, 1000
Atmospheric	Wind northward component	v	132	500, 850, 1000
Atmospheric	Geopotential	z	129	50, 500, 850, 1000
Static	Geopotential at surface (orography surrogate)	z	129	-
Static	Land sea mask	lsm	172	-
Generated	Top-of-atmosphere solar radiation	tisr	212	-
Generated	Latitude and longitude information	-	-	-
Generated	Day and year time information	-	-	-

Table 3.1: Channels used in training.

Data is stored on yearly NetCDF files, with a fixed number of temporal datapoints (1460). Each file has a size of approximately 118 GB.

Statistical data is generated using the train and test splits for both data normalization and ACC metric calculation. Climatology, computed on the period 1979-2017, is derived by taking the arithmetic mean for each grid point, channel, and time step within a year.

3.2 Codebase Adoption and Model Architecture

Due to the lack of published model weights for NVIDIA GraphCast, this project focuses on the Modulus codebase alone.

The codebase, comprising various scientific models, is cloned at commit `a5275d8`[1] (December 20th 2024), the repository HEAD during the code review.

To facilitate both reproducibility and the analysis of code contributions, the Modulus codebase is thoroughly vetted to identify the minimum set of required files, thus forming this project baseline. These include: model architecture, icosahedron and graph data structures, dataloader, and training script. Artifact generation is limited to simple image renderings for visualizing training progress. Inference facilities are absent.

Analysing this code thoroughly is essential for understanding how the original GraphCast publication was reimplemented into Modulus. This understanding helps identify possible areas of improvement. This effort is also expected to yield a practical understanding of the Message Passing mechanism of the GNN architecture.

Preliminary analysis indicates that significant refactoring is necessary to maximize training progress within the 30-minute debug queue (w.r.t., number of samples processed per unit

[1] `https://github.com/NVIDIA/modulus/commit/a5275d8`

of time), crucial for exposing the model to more data. In this project's context, more data improves results.

Meticulous attention is paid to how the data is loaded. All models should be trained using the same samples, presented in the same (shuffled) order. This careful data handling, performed through seed fixing, is essential for detecting subtle code improvements that might otherwise be missed due to random sampling.

Furthermore, variability in computational efficiency is expected through different model versions and training runs. Support for *Checkpoint-Restart* is thus needed beyond inference, allowing slower models to extend training past 30 minutes to align sample sequences seen by each model. The restart must ensure that the dataloader can resumes from the same point.

These measures contribute to providing equal learning opportunities and thus enable fair comparison of different model versions.

The original training script found in the Modulus codebase is written to work specifically with the data subset adopted for this project. FourCastNet's codebase [17] is thus also included for comparative purposes.

3.2.1 Learning Rate Schedule and Training Program

The model processes one data sample per rank (parallel process), resulting in a local batch size of 1. This represents one training iteration. Model weights are updated once per iteration.

Auto-regression (both for training and later for inference) is implemented as a rollout logic, feeding the model's output back as input. The number of rollout steps is increased through training. This creates sequences of time-aligned pairs between the reanalysis data (this project *ground truth*) and forecasts. These sequences, of length ≥ 1, are used for loss computation.

Longer rollout sequences aim at enhancing long-term prediction skills. Given the dataset 6-hours temporal resolution, a four-step rollout is required for a 1-day forecast.

The default training program in the codebase aligns with [22] and comprises three phases:

1. 1,000 iterations with a rollout length of 1. The learning rate linearly rises from 0 to 1e-3.
2. 299,000 iterations with a rollout length of 1. The learning rate decreases from 1e-3 to zero following a cosine curve.
3. 11,000 iterations with rollout length increasing by 1 every 1,000 iterations, ending with 12 rollout steps in the last 1,000 iterations. The learning rate remains fixed at 3e-7.

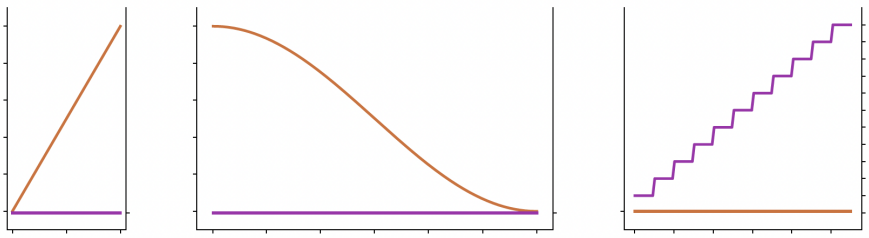


Figure 3.2: Image copied from [22]. Depiction of the three training phases. Purple represents the rollout length and orange the learning rate.

While all phases are vital for tuning model weights, the third phase is essential for building the forecasting skills on which evaluation will be conducted. The schedule thus needs to be adapted to allow the inclusion of rollouts > 1 within the runtime limits. This necessitates codebase refactoring aimed at increasing the training computational efficiency. Once the number of iterations per run is maximised, hyper-parameter tuning can optimize each phase length. See Section 3.4.

3.2.2 Loss Function and Optimizer

The original training objective is described in section 4.2 of [22]. The implementation in `https://vscode.dev/github/schuups/DSM500_FPR/blob/main/modulus-baseline/modulus/utils/graphcast/loss.py#L138` refines MSE by considering the data's domain. The loss is implemented as follows:

$$\mathrm{MSE}_{\mathrm{weighted}} = \frac{1}{C}\sum_{i=0}^{C} w_{\mathrm{latitude_i}} w_{\mathrm{variable_i}} \frac{(y_i - \hat{y}_i)^2}{\sigma^2_{\mathrm{temporal_i}}} \tag{3.1}$$

where:

- C is the number of channels in a sample, and $_i$ refers the the channel itself.
- $w_{\mathrm{latitude_i}}$ is the latitude weight, correcting spatial bias since grid cells near poles cover smaller areas than those near the equator due to Earth's spherical shape.
- $w_{\mathrm{variable_i}}$ is the per-variable normalization weight, ensuring each variable contributes equitably despite differing channel numbers (w.r.t., vertical levels).
- $\sigma^2_{\mathrm{temporal_i}}$ represents temporal variance. Inversely weighting this variance prioritizes channels with lower variability, favouring more predictable and stable phenomena.
- y_i is the predicted and $\hat{y}_i$ the expected channel value.

The key differences with the original publication are in the mean taking at both batch and rollout levels. The PyTorch *DistributedDataParallel* module handles batch-level mean transparently if batch_size stays 1, but the rollout-level mean is implemented differently. Defined in `https://vscode.dev/github/schuups/DSM500_FPR/blob/main/modulus-baseline/train_base.py#L24`, it takes the sum of rollout step-level losses before executing the backward pass.

The optimizer (AdamW) and gradient norm clipping follow the publication.

3.2.3 Reproducibility Instructions

This project code, including notebooks, execution scripts, and reproducibility-oriented setup instructions, is available under `https://github.com/schuups/DSM500_FPR`.

3.3 Results Evaluation Protocol

The evaluation protocol is discussed before the experimental approach to guide its design and optimize refinement iterations.

Storage limitations prevent large volumes of inference output from being kept. Outputs are processed swiftly to free up space. Key details of interest like metrics to compute and artifacts to render are defined in this section.

3.3.1 Metrics Reported

Following Chapter 2, the metrics of interest are MSE and ACC. These are collected globally and for the headline variables `z500`, `t850`, `t2m`, `msl`, `10u`, `10v`, `u850` and `v850`.

To support regional evaluation of results, a selection of ECMWF scorecards' regions (see Figure 3.3) is adopted: `n.hem`, `tropics`, `s.hem`, `europe`, `n.amer`, `e.asia` and `austnz`. This selection balances hemispheric coverage within practical presentation limits.

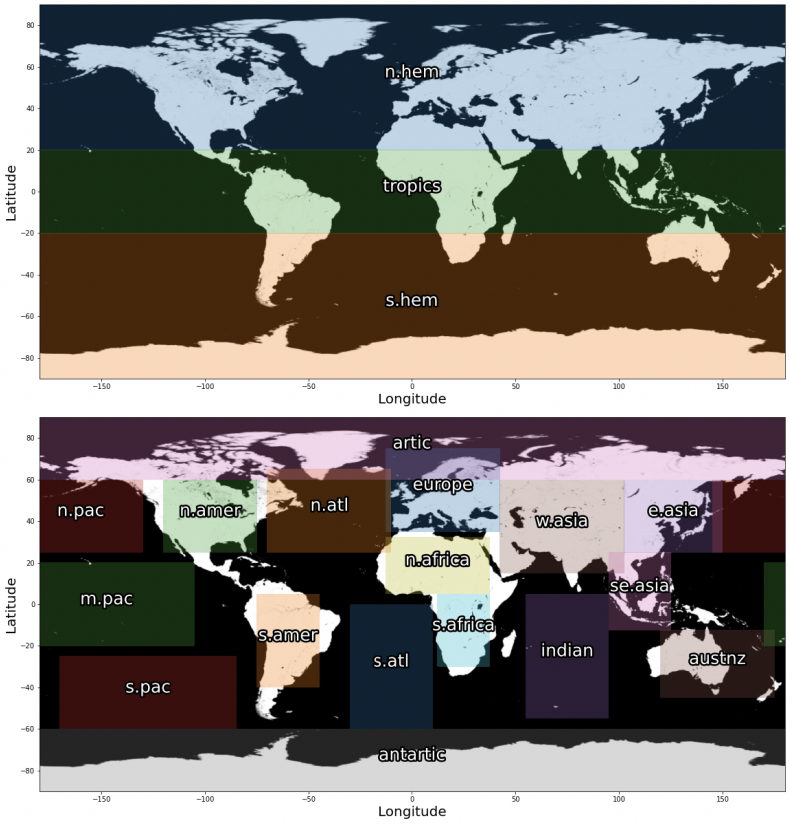


Figure 3.3: Image copied from [22]. Depiction of the regional areas and their names.

To support comparison across lead times, metrics are reported for each of the above-described slices on lead times up to 7 days. For lead-time-specific comparisons, 1, 3 and 7 days are adopted as comparison points. These are shorter than what [22] adopted due to the much restricted training time.

There are thus 4'480 measurements to be collected for each model being evaluated, i.e., 2 metrics, for 10 variables (9 headline, plus 1 global), for 8 regions (7 regions, plus 1 global), and 28 time steps (7 days).

3.3.2 Results Collection

The evaluations are conducted on the out-of-sample split (i.e., year 2018), comprising 1'460 samples. Given 28 time steps required for the maximum lead-time of interest (7 days), the useful samples are 1'431.

With 27 well-spaced initial conditions, the uniformity of daily and yearly coverage can be maximized. These forecasting trajectories are obtained for each training that can be executed on a given model (w.r.t. the quantity of iteration-equivalent checkpoints provided for testing).

Furthermore, to reduce the bias that a particular training initialization seed has on a training outcome, multiple training runs are executed on different seeds and the collected metrics averaged arithmetically. Accounting for the seeds tested, and initial conditions employed, 81 samples are collected for each metric measure taken.

3.3.3 Baselines and Performances Recordings

Metrics are reported for several versions of GraphCast, e.g. a baseline and improved versions.

Additionally, the reliance of this project on the same dataset used by FourCastNet [17] facilitates the inclusion of FourCastNet as another baseline. If project time allows, a FourCastNet model is trained from scratch using the same protocol applied for this project. Published weights are not used due to differences in training sample sequences and training runtime, which prevent meaningful comparisons.

3.3.4 Results Presentation and Visualization

Results are presented through a variety of data products:

- Grids of comparative line plots, of both RMSE and ACC, similar to those found in [22].
- Scorecards similar to those by ECMWF available under `https://sites.ecmwf.int/aifs/scorecards/scorecard_aifsv021_ifs.html` or by [38].
- Channel renderings to effectively illustrate forecast development over rollout windows.

3.3.5 Statistical Comparison

Due to the relatively small sample size and the bounded, non-symmetric nature of evaluation metrics such as ACC and RMSE, it is assumed that the distribution of results is non-normal. Consequently, statistical significance is assessed through pairwise comparisons between the baseline and improved models using the non-parametric Wilcoxon signed-rank test. To complement hypothesis testing and assess the practical relevance of observed differences, effect size is provided using Cohen's d.

3.4 Experimental Approach

The main goal of trainings is to minimize testing MSE on unseen data. Progress is monitored via *wandb*[2] for early detection of situations such as over-fitting, instability, vanishing or exploding gradients, and generalization gaps.

The process followed to obtain the results is so structured:

1. A **GraphCast dry-run** is executed to assess the functioning of the Modulus codebase (w.r.t. missing dependencies). A lower-bound benchmark is established with respect to the minimum number of samples that a model can process within the runtime limit.
2. An **inference framework** is developed to collect post-training results so to implement the protocol described in Section 3.3. This development effort continues throughout the experimentation phase.
3. **Refactoring effort** is invested to meet the evaluation protocol requirements:
   - Support skill performances comparison by having a reproducible training process, centred around seed fixing and predictable dataloader sampling.
   - Improve computational efficiency to maximise number of samples processed in the runtime limit.
4. A **GraphCast performance baseline** is established using the refactored code, on which the model architecture is still equivalent to the original version by NVIDIA. This serves as a critical reference for measuring improvements on it.
5. Hyperparameters are systematically tuned to identify an **optimal training schedule and establish the experiments iteration count**. This leads to defining the optimal duration of each of the three distinct training phases, and the number of iterations to run the training for (w.r.t. to earlier sections) to support fair comparative evaluation.
6. **Iterative enhancements** are implemented that adapt and introduce new features to GraphCast. Attention is paid to implement improvements as optional (via toggles) so as to enable ablation studies later on.
7. An **ablation study** is performed through systematic hyperparameter exploration using *wandb*'s Sweeps[3], enabling the assessment of individual contributions to model performances. The best possible configuration of GraphCast w.r.t. the implemented changes, is trained to form an improved model and a performance measurement is taken of it. This serves as the basis for the comparative performances analysis against the baseline.
8. Should sufficient time remain, a **FourCastNet** model is trained on the same data used for the GraphCast models training and incorporated for additional comparative evaluation, broadening the performance context against established benchmarks.
9. Finally, the necessary is developed for **effective results interpretation**, e.g. as grid plots (and videos, if time allows) on users-relevant metrics.

[2] *Weights & Biases* is a developer platform for building AI application, `https://wandb.ai/`
[3] `https://docs.wandb.ai/guides/sweeps`

3.5 Infrastructure

The infrastructure utilized is the *Alps*[4] RI hosted by the Swiss National Supercomputing Centre (CSCS), located in Lugano, Switzerland. At the time of writing, *Alps* is ranked as the 7th most powerful supercomputer worldwide[5]. This HPC cluster can be utilised through SLURM, a queue-based workload manager. An application execution is packaged as job script and placed in a SLURM queue. Execution starts once sufficient resources are available.

Each *Alps* node is equipped with four NVIDIA GH200 superchips, each featuring 72 Grace ARM CPU cores, 128 GB LPDDR 5X RAM, and one NVIDIA H100 GPU with 96 GB HBM3 memory. The four GPUs within a node use NVLink for communication, while inter-node communication happens via HPE Cray Slingshot-11 at 200 Gbps per GPU module.

The project working files are stored on the *Alps*' Lustre file-system, which is backed by a high-performance full-flash storage system.

Alps is tailored for the usage of application containers. This project adopts the NVIDIA Modulus container version 24.09[6] from the NVIDIA's NGC catalog. Additional dependencies, documented in the reproducibility documentation[7] are managed via virtual environment.

This project predominantly uses nodes from the debug queue, which is configured to limit runtimes to 30-minute and allocation sizes up to 2 exclusive nodes (8 GPUS).

Trainings are executed as single-node jobs, each running four parallel training processes (one per GPU). Training parallelization is performed only on the data axis, where each rank independently processes distinct samples. Gradients are computed independently by each rank and synchronized via an *AllReduce* operation during the backward pass. Once all ranks have the same set of gradients, weights are updated independently. The result is that after each backward pass each training process holds an identical model copy.

Weights & Biases is used for training progress observability and also for the hyper-parameter optimization execution (*Sweeps*[8]).

The project codebase is version-controlled on GitHub `https://github.com/schuups/DSM500_FPR`.

[4] `https://www.cscs.ch/computers/alps`
[5] `https://top500.org/lists/top500/2024/11`
[6] Container tag: `nvcr.io/nvidia/modulus/modulus:24.09`
[7] `https://github.com/schuups/DSM500_FPR/blob/main/REPRODUCIBILITY.md`
[8] `https://docs.wandb.ai/guides/sweeps/`

4. Results

4.1 Modulus' GraphCast Dry-Run

The dry-run verifies logic, configuration, and dependencies to ensure codebase viability.

The identified missing dependencies include: (1) statistical files for normalisation, (2) time-wise standard deviations for the loss function inverse variance weights, (3) channels meta-data for loss function variable weights, and (4) static data download for land-sea masks and geopotential. Creating these required significant effort, sometimes involving reverse engineering. Detailed instructions available under `https://github.com/schuups/DSM500_FPR/tree/main/supplementary_files/dry-run`.

Two configurations are tested: (1) the initial *out-of-the-box* setup with fixed dependencies and (2) an adjusted setup with minor improvements. These minor improvements include (1) reducing CPU-based rendering frequency of testing images from every 5 to every 100 iterations, and (2) reducing checkpointing frequency from every to every 50 iterations. Each configuration was executed twice as soft-validation.

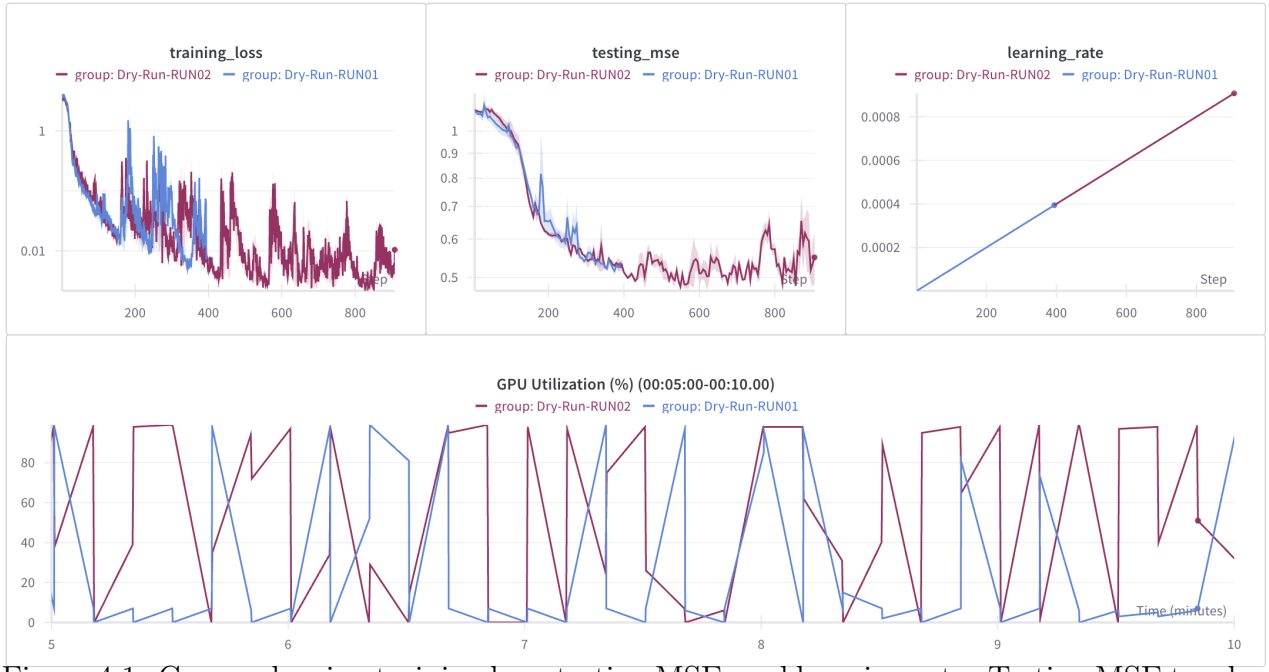


Figure 4.1: Curves showing training loss, testing MSE, and learning rate. Testing MSE touches on the .5 level. Run details available at `https://api.wandb.ai/links/schups/dlorz5s4`.

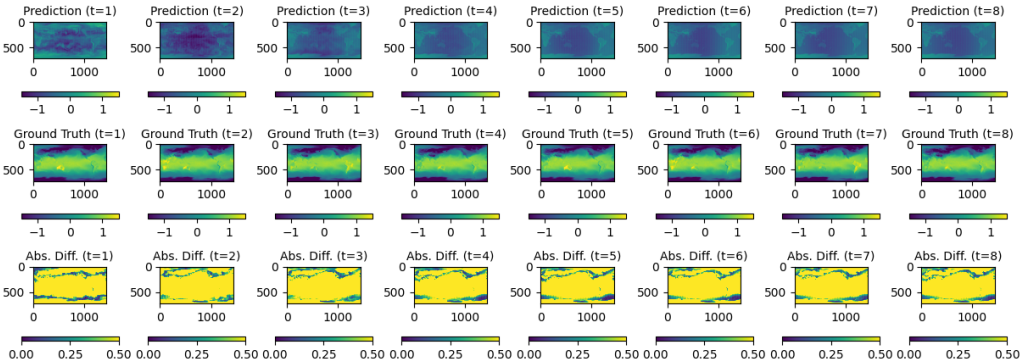


Figure 4.2: Visualization of testing rollout (t2m variable) at iteration 5, as generated by the original script version.

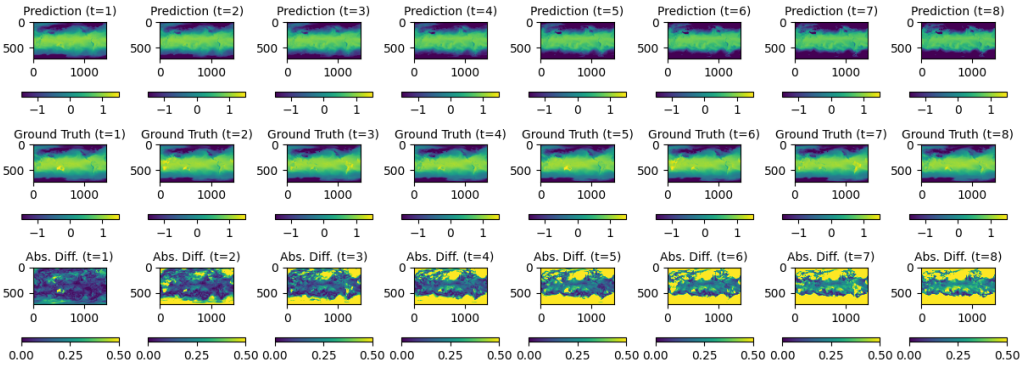


Figure 4.3: Visualization of testing rollout (t2m variable) at iteration 900, as generated by the original script version.

Comments

- Within the 30-minute limit, the default configuration achieves around 390 iterations; the trivially-improved configuration reaches around 900 iterations. Given that *batch_size* is fixed to 1 by design, that equates to number of samples seen by each GPU.
- Training curves appear overall healthy: both show decreasing trends with the longer run showing lower values. A slight increase at the end of the longer run's testing MSE might seem a sign of over-fitting, yet the training loss around those iterations experience similar spikes. This might be due to higher learning rates causing instability.
- GPU utilization shows intermittent idle periods (w.r.t., linked report).
- The first rows on the test plots in Figures 4.2 and 4.3 demonstrate the improved autoregressive prediction skill of the model.
- Code analysis reveals that testing loss calculation uses pure MSE - thus without the weights used for the training loss. The training and testing values (y-axis) are therefore not directly comparable.

4.2 Inference Framework

Comparing models for forecasting abilities is this project's main focus. Chapter 3 outlines the protocol, and this section details its implementation in an inference framework.

For readability purposes, the term *model version* is subsequently used to refer to the pair of both model version (code) and its hyper-parameters.

To fairly evaluate different model versions, the following aspects are considered:

1. All models are trained from scratch on the same data, i.e. the same amount of samples, the same specific samples, processed in the same sequential order. While the training dataloader should still produce shuffled data (e.g. to allow the model to see data from all seasons), its sampler must support predictability configurations.
2. To mitigate the effect that model initialization (i.e., initial weights) has on obtained predictive abilities, each model version is trained multiple times by fixing the randomization seeds to different values. Therefore, for each model version evaluated, multiple checkpoints are provided, one for each seed. Furthermore, such model checkpoints are aligned on the iteration number at which they are saved.
3. Given the 30-minute runtime limit and the varying computational efficiencies (not all models train at the same speed), processing the same amount of data per training session requires the *checkpoint/restart* functionality to include the data loader, so that sample sequences can be resumed deterministically.
4. Once trained, models evaluation need to use *out-of-sample* data which fairly represents various seasons and times of day (e.g., daytime, nighttime). This ensures that models strong in specific situations (e.g., winter) are not unfairly advantaged. While not all models use the same channels (e.g., `sst`), they should all be tested on the same samples.
5. For a more nuanced assessment of model abilities, evaluation should be possible at variable, geographical, and lead time levels. The output of the inference should retain sufficient information.

The inference framework, available under the GitHub repository is designed to process configuration files such as the following.

```
inference:
  rollout_steps: 28
  data:
    metadata: ./DSM500_FPR/data/FCN_ERA5_data_v0/metadata.yaml
    file_paths:
    - ./DSM500_FPR/data/FCN_ERA5_data_v0/out_of_sample/2018.h5
    initial_conditions_idx: [0, 55, 110, 165, 220, ..., 1210, 1265, 1320, 1375, 1430]

models:
- name: gc-baseline-run01
  type: gc-baseline
  code_path: ./DSM500_FPR/modulus-baseline
  config_path: ./DSM500_FPR/modulus-baseline/conf/config.yaml
  weights:
  - ./DSM500_FPR/modulus-baseline/runs/RUN01/SEED21/checkpoint/model.iter003750.pth
  - ./DSM500_FPR/modulus-baseline/runs/RUN01/SEED42/checkpoint/model.iter003750.pth
  - ./DSM500_FPR/modulus-baseline/runs/RUN01/SEED84/checkpoint/model.iter003750.pth

- name: gc-improved-run02
  type: gc-improved
  code_path: ./DSM500_FPR/modulus-improved
  config_path: ./DSM500_FPR/modulus-improved/conf/config_tuned_config.yaml
  weights:
```

```
24   - ./DSM500_FPR/modulus-improved/runs/RUN02/SEED21/checkpoint/model.iter003750.pth
25   - ./DSM500_FPR/modulus-improved/runs/RUN02/SEED42/checkpoint/model.iter003750.pth
26   - ./DSM500_FPR/modulus-improved/runs/RUN02/SEED84/checkpoint/model.iter003750.pth
27
28 - name: fourcastnet
29   type: fcn
30   code_path: ./DSM500_FPR/fourcastnet-92260c1
31   config_path: ./DSM500_FPR/fourcastnet-92260c1/config/AFNO.yaml
32   weights:
33   - ./DSM500_FPR/fourcastnet-92260c1/checkpoints/iter3750.pth
34
35 metrics.slices:
36   variables:
37   - name: all
38   - name: z_500
39   - name: t_850
40   - name: t2m
41   - name: msl
42   - name: 10u
43   - name: 10v
44
45   regions:
46   - name: global
47   - name: n.hem
48     latitude_range: [20, 90]
49   - name: tropics
50     latitude_range: [-20, 20]
51   - name: s.hem
52     latitude_range: [-90, -20]
53   - name: europe
54     latitude_range: [35, 70]
55     longitude_range: [-12.5, 42.5]
56   - name: e.asia
57     latitude_range: [25, 60]
58     longitude_range: [102.5, 150]
```

Listing 4.1: An inference framework configuration file example.

Further to the above,

- Metrics are calculated for each of the 28 lead time steps (`inference.rollout_steps`), allowing review throughout the forecast period. Due to limited training runtime, models are expected to perform better at shorter lead times.
- The default initial condition indices `inference.data.initial_conditions_idx` (27 items, truncated in the example for formatting reasons) are chosen to balance representation of seasons through out the year, as well as time of day (i.e., both daytime and nighttime samples) while minimizing the quantity of data produced and which need to be stored.
- Multiple model weights (checkpoints) can be provided for each model, representing training runs initialized with different random seed values, but captured at the same iteration.
- Slices of interest can be defined for variables and regional areas. The framework permutes them, computing metrics for each pair. The regional boundaries are defined following ECMWF scorecards, of which `https://sites.ecmwf.int/ifs/scorecards/scorecards-49r1HRES.html` is an example.
- The special slices `all` and `global` are used to select all variables or the entire globe area.
- The metrics (RMSE and ACC) are left implicit, and they are always computed. The computation of climatological data needed for the ACC metric calculation is available in the GitHub repository.

```python
activities = build_activities(
    cfg=OmegaConf.load("supplementary_files/inference/conf/config.yaml")
)

print("Activities to run:", len(activities))
print("An activity:", activities[5])
```

✓ 0.0s Python

```
Activities to run: 189
An activity:
        Activity ID:       5
        Model name:        gc-baseline-run01
        Model type:        gc-baseline
        Model weights:     ./DSM500_FPR/modulus-baseline/runs/RUN01/SEED21/checkpoint/model.iter003750.pth
        Initial condition: 275
```

```python
df
```

✓ 0.0s Python

	activity_id	name	type	weights	variable	region	metric	step	value
0	0	gc-baseline-run01	gc-baseline	./modulus-baseline/SEED21/model.iter003750.pth	all	global	rmse	0	0.000000
1	0	gc-baseline-run01	gc-baseline	./modulus-baseline/SEED21/model.iter003750.pth	all	global	rmse	1	1152.000000
2	0	gc-baseline-run01	gc-baseline	./modulus-baseline/SEED21/model.iter003750.pth	all	global	rmse	2	1288.000000
3	0	gc-baseline-run01	gc-baseline	./modulus-baseline/SEED21/model.iter003750.pth	all	global	rmse	3	1344.000000
4	0	gc-baseline-run01	gc-baseline	./modulus-baseline/SEED21/model.iter003750.pth	all	global	rmse	4	1400.000000
...	...	...	...	...	...	...	...	...	...
762043	188	fourcastnet	fcn	./fourcastnet-92260c1s/iter3750.pth	v_850	austnz	acc	23	0.995642
762044	188	fourcastnet	fcn	./fourcastnet-92260c1s/iter3750.pth	v_850	austnz	acc	24	0.996500
762045	188	fourcastnet	fcn	./fourcastnet-92260c1s/iter3750.pth	v_850	austnz	acc	25	0.995884
762046	188	fourcastnet	fcn	./fourcastnet-92260c1s/iter3750.pth	v_850	austnz	acc	26	0.996870
762047	188	fourcastnet	fcn	./fourcastnet-92260c1s/iter3750.pth	v_850	austnz	acc	27	0.995914

762048 rows × 9 columns

```python
df_subset = df[(df.metric == "acc") & (df.variable == "t2m") & (df.region == "austnz")]

with sns.axes_style("darkgrid"):
    ax = sns.relplot(data=df_subset, x="step", y="value", hue="label")
    ax.figure.suptitle("ACC score over forcast lead time steps (variable: t2m, region: austnz)")
    ax.set_axis_labels("Lead time steps", "ACC")
```

✓ 1.0s Python

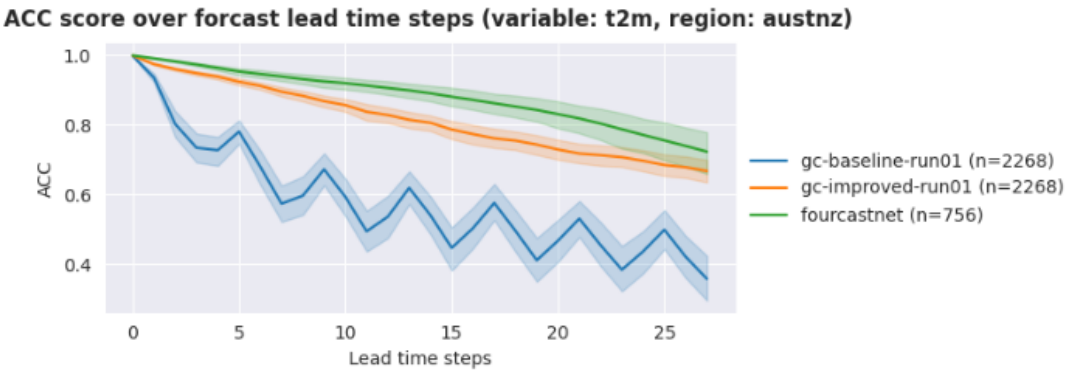


Figure 4.4: Depiction of pipeline elements such as (1) inference activities to be executed, (2) raw results collected, and (3) results visualization.

Each combination of model, weight, and initial condition define an inference job (called *Activity*). Each activity produces metrics for all defined slices, generating kilobyte-sized results for line plots, and gigabyte-sized forecasts needed for images rendering. Metrics for a particular model are collected for each of its weights and for each initial condition (w.r.t. `n` in the legend above). This enables spread visualization, assisting in model reliability assessment.

4.3 Code Base Analysis and Refactoring

Modulus is a comprehensive library that includes a wide range of physics-informed ML models, among which GraphCast is only one. Given the scale and complexity of the codebase, reducing comprehension overhead early on is considered key.

Initially, pruning is performed to isolate the minimal subset of files needed to operate the GraphCast model, resulting in the `modulus-dry-run` version. A detailed code review and extensive refactoring are performed on it, producing the `modulus-baseline` version.

This activity is guided by the following objectives:

- **Gain a broad understanding** of the codebase and its functionality to (1) contain the cognitive load associated with in-depth handling of someone else project (given the lack of an API and documentation), and (2) make development iterations faster and robust.
- **Enable determinism** in data loading, checkpointing, and training - essential for (1) reproducibility and consistency as required by the evaluation protocol, and to (2) validate that computational efficiency optimizations preserve the original functional behaviour.
- **Optimize computational efficiency**, to increase the data throughput within the jobs runtime limit, i.e., number of training iterations within 30 minutes.
- **Reduce peak GPU memory footprint**, to enable the exploration of a broader hyper-parameter space (e.g., deeper networks or larger hidden layers) later in the project.

Key results are summarized as follows:

- Dataloading and model checkpointing are refactored to support both random seed fixing and restart offsets, enabling the continuation of sampling sequences.
- Significant gains in computational efficiency increased the number of iterations processed within the same wall-time from approx. 900 (`modulus-dry-run`) to approx. 4000. This is achieved through several optimizations, among which:
  - Startup latency is reduced by caching the graph data structure (otherwise created anew every time) and enabling earlier data serving by the dataloader.
  - Activations checkpointing is refactored (from its original configuration) to adjust dynamically during training, affecting gradually more layers when available GPU memory is expected to become scarcer. This happens mainly due to larger back-propagation graphs on longer roll-outs. This adaptive strategy, based on heuristics developed early in the project, shifts the memory-computation trade-off over time, taking advantage of available GPU memory when this is abundant.
  - The computation of derived input channels, such as the top-of-atmosphere solar irradiance, is moved from the main CPU thread to the NVIDIA DALI sub-threads.
  - Where feasible, intermediate computations are unified under consistent precisions to reduce type conversions. Tensors allocated to the GPU are kept at precision levels congruent with that of the training, and not higher. Trivial opportunities are exploited both for data movement reductions and for fusion of CUDA kernels.
  - Gradients norm clipping was removed from the training loop in favour of monitoring values externally through Weight and Biases dashboards.

- Multi-step autoregressive training, previously non-functional due to implementation bugs, was re-enabled. Although the fix was conceptually simple, it exposed new GPU memory bottlenecks related to growing autograd graphs on longer roll-outs. The solution adopted can be reviewed by comparing the original version with the new one.

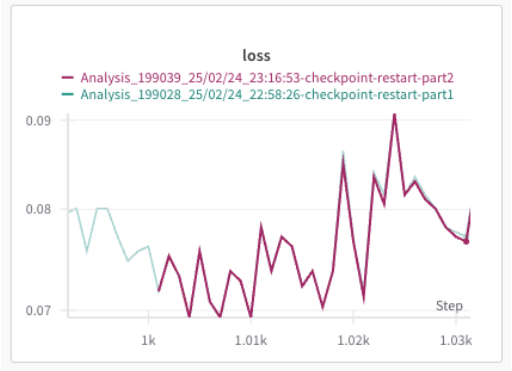


Figure 4.5: Example of training resumption from model checkpoint, demonstrating the overlap of the subsequent training steps' loss values.

The refactoring can be reviewed in detail by comparing folders `modulus-dry-run` and `modulus-baseline`. The next section focuses on the computational performance gain achieved.

4.4 Performance Baseline

On unchanged hyper-parameters and equal runtime limit, the refactored code boosts training throughput from 900 to approx. 4,000 iterations, while brining GPU utilization to a high stable level. Compared to the dry-run version, the curves appear more stable and converging. The longer training run results in better generalization (i.e., lower testing MSE levels).

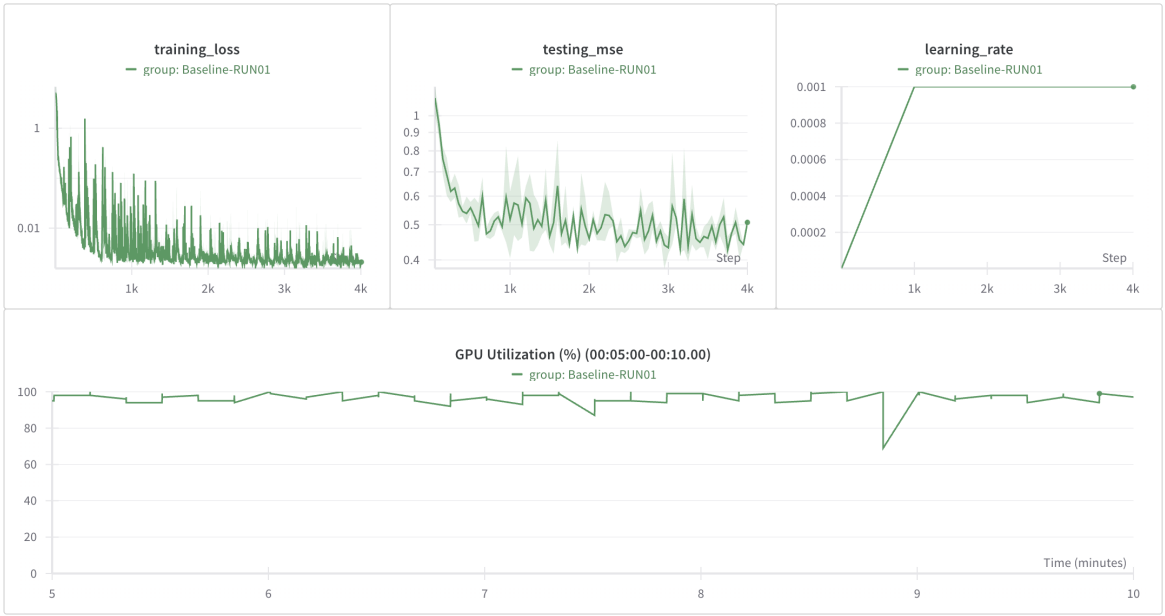


Figure 4.6: The longer training run allows the testing MSE to break below the .5 level. Run details available at `https://api.wandb.ai/links/schups/8bz1dayi`.

4.5 Optimal Learning Schedule

Regarding the training schedule, the Modulus implementation default hyperparameters align with those used in the original DeepMind publication. However, these do not account for this project's 30-minute runtime constraint. Under such constraint, only the first phase (default: 1000 iterations) and a negligible part of the second (default: 299'000 iterations) are executed.

Given the model's autoregressive nature, it is plausible that later phases (given longer roll-outs) significantly contribute to performance gains. The default schedule is thus assumed to be suboptimal for this project's context.

With the goal of identifying a training schedule configuration that consistently maximizes model skill gains within the runtime limit, a hyperparameter tuning task is performed. Experiments are orchestrated via *wandb*'s Sweeps[1] functionality. The randomized search strategy is used, instead of a grid search, to cope with the limited compute. The random seed hyperparameter is included in the search with the purpose of minimizing bias from models initialization. The search objective is to minimize testing MSE.

Figure 4.7: Learning curves of about 101 runs involved in the training schedule hyperparameters optimization. The testing MSE .4 level is surpassed. Experimental details are available at `https://api.wandb.ai/links/schups/gzn88h8n` and in the GitHub repo.

Data collected is filtered by runtime (> 25 minutes) and by testing MSE level reached (< 0.45), resulting in 19 samples. See Figure 4.8. The following conclusions are drawn:

- The second training phase is the most relevant to achieve better testing MSE levels, followed by the first training phase.
- Learning rates for the first and second phases need to be increased from 1e-3 and 3e-7, to 5e-3 and 5e-3 respectively.
- Training should include at least two rollout increments.

Further analysis of results reveals the extent of the training throughput slowdown caused by longer roll-outs. To accommodate the roll-out increments suggested by the optimization just performed, while keeping the iteration count consistent with that of the baseline, one model checkpoint restart is permitted. This compromise is done to respect the defined experimentation protocol so as to enable fair results comparison. This also means that training will need two jobs, extending beyond the runtime constraint of 30 minutes thanks to the implemented fully deterministic checkpoint restart functionality.

[1] `https://docs.wandb.ai/guides/sweeps`

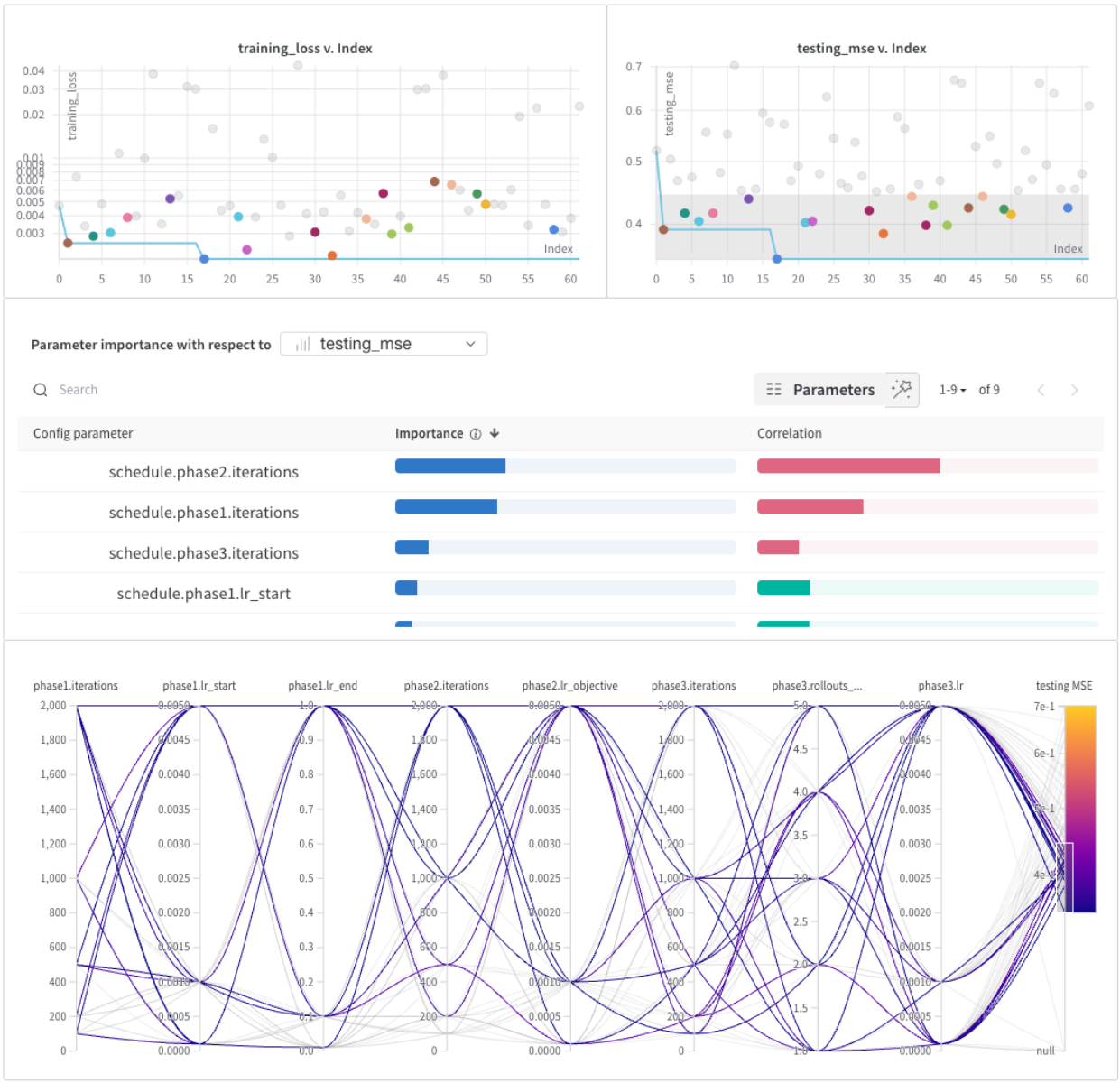


Figure 4.8: Visualization and correlation of the randomized hyper-parameters search results. On the upper panels, each datapoint represents a separate model training. Details at `https://api.wandb.ai/links/schups/gzn88h8n`

The sweep configuration can be reviewed at `https://github.com/schuups/DSM500_FPR/blob/main/supplementary_files/schedule_tuning/sweep.yaml`.

4.6 GraphCast Enhancements

This section gives an overview of several key enhancements that have been implemented in the codebase to improve GraphCast predictive capabilities.

Their impact on GraphCast's predictive performances is analysed in Section 4.7. To enable the ablation study, the activation of proposed enhancements is controlled via toggle switches.

4.6.1 Sea-Surface Temperature Effect on Inverse Variance Weights

Although part of the dataset, the sea-surface temperature (`sst`) channel was not used to train FourCastNet [17][2]. Yet, the statistics needed for its normalization are included[3].

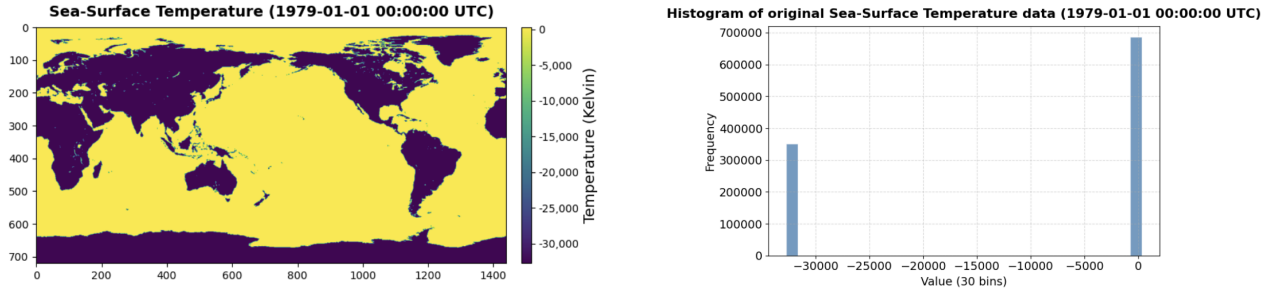


Figure 4.9: Visualizations of `sst` data, both as image and as values frequency distributions.

The `sst` data, expressed in Kelvin degrees, exhibit a mean value inconsistent with the physically plausible lower bound of the property it aims to describe[4]. By its definition, the channel lacks land-based data, which is filled with the sentinel value $-32'767$, which as described in [39] is a convention in meteorological datasets.

When inverse variance weighting is applied in the loss computation, lower-variance channels like `sst` receive disproportionately high weights, causing them to dominate the loss function.

It is thus proposed that instead of excluding this channel from the training (as done by [17]), the filling value is replaced with semantically similar data from the `t2m` (near-surface temperature) channel. Normalization statistics are recomputed accordingly.

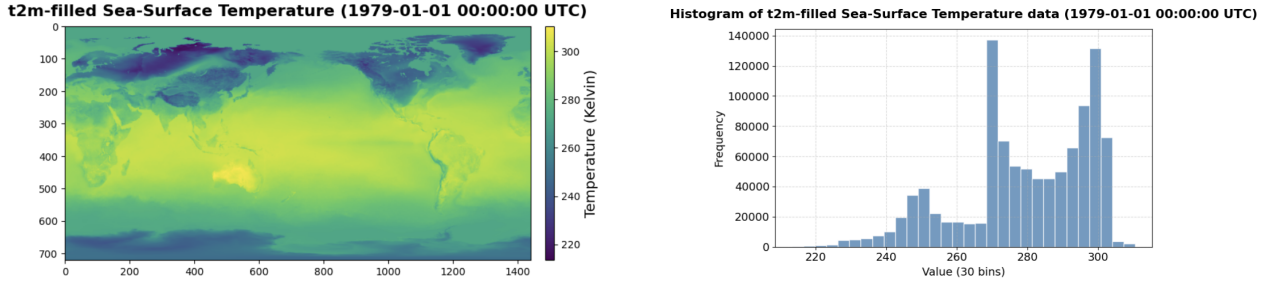


Figure 4.10: Visualizations of the `sst` channel after replacing the sentinel values.

Figure 4.11 shows the inverse variance weights before and after the proposed enhancement.

[2]Recall, the dataset used for this project is the same employed for training of FourCastNet.
[3]See `global_means.npy` and `global_stds.npy` under `https://github.com/NVlabs/FourCastNet`.
[4]The absolute zero is 0 °K.

Channel:	Old weight:	New weight:	Channel name:
0	0.4357	0.4357	U-Wind Component (at 10 meters)
1	0.3839	0.3839	V-Wind Component (at 10 meters)
2	0.3155	0.3155	Air Temperature (at 2 meters)
3	0.0039	0.0039	Surface Pressure
4	0.0037	0.0037	Mean Sea Level Pressure
5	0.6005	0.6005	Temperature (at 850 hPa)
6	0.3956	0.3956	U-Wind Component (at 1000 hPa)
7	0.3456	0.3456	V-Wind Component (at 1000 hPa)
8	0.0046	0.0046	Geopotential (at 1000 hPa)
9	0.3180	0.3180	U-Wind Component (at 850 hPa)
10	0.2844	0.2844	V-Wind Component (at 850 hPa)
11	0.0051	0.0051	Geopotential (at 850 hPa)
12	0.2536	0.2536	U-Wind Component (at 500 hPa)
13	0.2097	0.2097	V-Wind Component (at 500 hPa)
14	0.0042	0.0042	Geopotential (at 500 hPa)
15	0.7400	0.7400	Temperature (at 500 hPa)
16	0.0042	0.0042	Geopotential (at 50 hPa)
17	0.0441	0.0441	Relative Humidity (at 500 hPa)
18	0.0545	0.0545	Relative Humidity (at 850 hPa)
19	0.2881	0.2881	Total Column Water Vapour
20	11.7701	0.3239	Sea Surface Temperature

Figure 4.11: Comparison of old and new loss function inverse variance weights.

This enhancement duplicates some data, which could be problematic. The loss weights of the temperature variables group have been adjusted to account for this. Additionally, a toggle feature allows inclusion or exclusion of the `sst` channel.

4.6.2 Improved Smoothness for Temporal Information

Table 3.1 illustrates the channels dynamically generated by the dataloader during training to provide additional contextual signals. Following the original DeepMind implementation, four channels encode each sample's temporal position in the annual and diurnal cycles.

However, code analysis reveals seemingly suboptimal handling of cyclical information. Transitions like December 31 to January 1 and nighttime to morning were not adjacent, appearing as distant samples in encoded space. This misrepresentation may hinder the model's ability to learn smooth, periodic patterns inherent in the underlying physical processes.

The proposed enhancement smooths such transitions. Figure 4.13 illustrates. Furthermore, it also improves temporal information quality at a localized level, as shown in Figure 4.14. Some single-step loss spikes, exemplified in Figure 4.12, are found to be caused by the mishandling of temporal information at such a local level, causing continuity gaps in lap years.

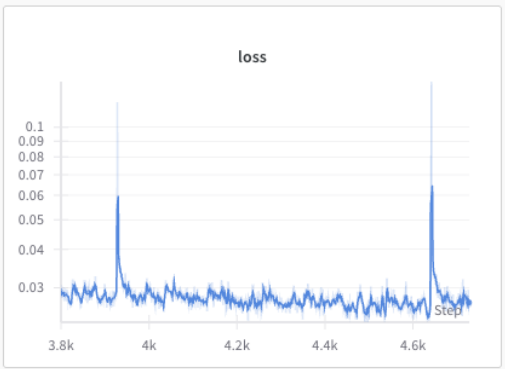
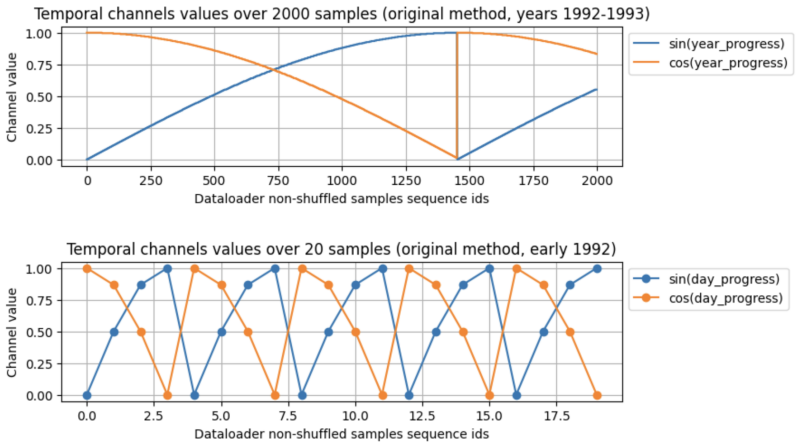
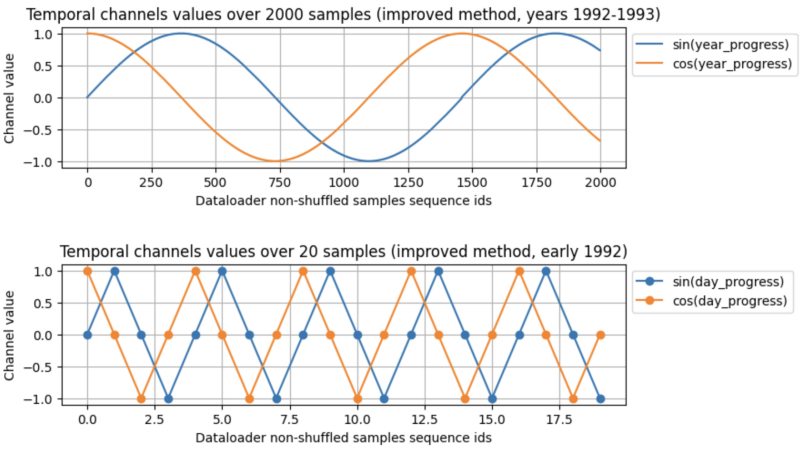


Figure 4.12: Examples of single-step loss spikes, some of which are found to be caused by step value changes in temporal information channels on lap years related differences.

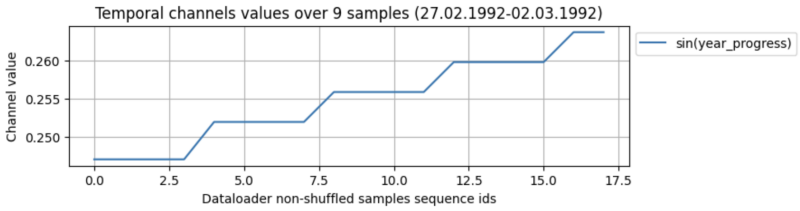


(a) Temporal information as originally implemented.

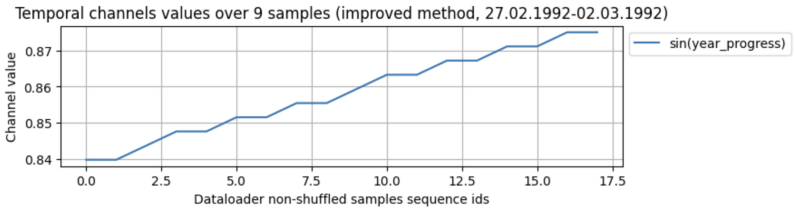


(b) Temporal information, through the proposed improvement.

Figure 4.13: Visualization of how temporal data changes across an ordered sequence of samples.



(a) In the original implementation, annual cycle progression values are fixed within the diurnal cycle.



(b) The proposed improvement smoothen such step changes.

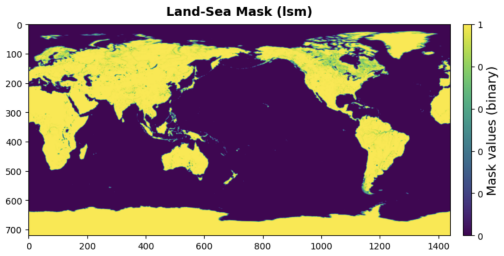
Figure 4.14: Information content improvement at local level on the yearly cycle data.

4.6.3 Misalignment of Top-of-Atmosphere Solar Irradiance

The dataloader dynamically generates the top-of-atmosphere (ToA) solar irradiance channel, whose aim it to approximate solar exposure above cloud level. However, detailed analysis of the implementation revealed a misalignment between the longitude ranges used in the irradiance computation and the actual spatial layout of the dataset. This discrepancy introduces spatial inconsistencies in the irradiance values generated. Figure 4.15 illustrates.

```
296
297         # create the lat_lon_grid for the cos_zenith computation
298         self.latitudes = torch.linspace(-90, 90, steps=input_res[0])
299         self.longitudes = torch.linspace(-180, 180, steps=input_res[1] + 1)[1:]
300         self.lat_lon_grid = torch.stack(
301             torch.meshgrid(self.latitudes, self.longitudes, indexing="ij"), dim=-1
302         )
303
```

(a) Code responsible for the channel generation inputs.

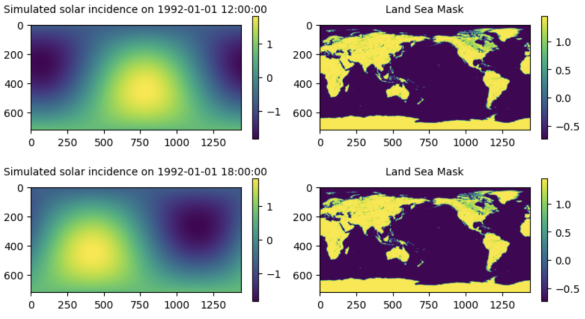


(b) Dataset longitudinal spatial layout.

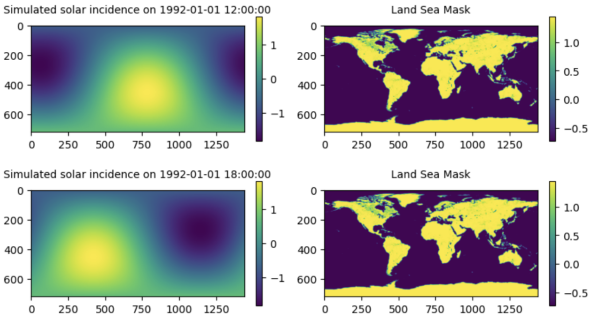
Figure 4.15: Misalignment between data used to generate a dynamic channel and the actual layout of longitudes in the dataset.

The range ($-180°$ to $180°$) used for the generation of channels such as ToA solar irradiance is misaligned with the range on which data is layed out on the dataset ($0°$ to $360°$).

Figure 4.16 illustrates how such misalignment leads to solar irradiation peaking in geographical areas facing the night, potentially obstructing the model's ability to learn periodic patterns in the physical processes it aims to model.



(a) Original implementation.



(b) Enhanced implementation.

Figure 4.16: Comparison of effect on channel data generated from the (a) original implementation and the (b) proposed implementations. Times displayed on plot titles are UTC.

The proposed improvement aligns the prime meridian with the dataset's spatial center.

4.6.4 Seasonal Bias Removal

From the storage perspective, data is organized in yearly buckets, with each year stored in a separate file of approximately 118 GB. Code analysis revealed that, to accommodate rollout

lengths, the sampling logic restricted the range of valid start indices, effectively excluding samples that span across year boundaries. As a result, data from late December was systematically under-represented in all training and evaluation phases.

To address this seasonal bias, the dataloader was refactored to support cross-file sampling. When a sample starting in late December is selected, the dataloader now seamlessly collates data from the subsequent year's file. This correction increases the number of valid available samples from 53,947 to 54,019 (for rollout length equal to 1), brining uniform temporal coverage throughout the year.

4.7 Ablation Study

The proposed enhancements discussed so far are assumed to improve the predictive abilities of GraphCast. However, isolating their individual contributions is computationally challenging because of the complexity of evaluating all dependency combinations.

To selectively evaluate these features, feature toggles were implemented. These are used to perform an ablation study through a randomized search of toggle combinations. This is orchestrated via *wandb*'s Sweeps. The sweep configuration is available at `https://github.com/schuups/DSM500_FPR/blob/main/supplementary_files/ablation_study/sweep.yaml`.

A total of 137 runs are launched, 108 of which reach a runtime above 25 minutes. Filtering for runs achieving a test MSE below .375 (i.e., better than previously achieved), yields 44 promising configurations. Due to the runtime limit and the inclusion of rollouts longer than 1, trainings are typically terminated by SLURM (the cluster's workload manager) around iteration 2'500. This is also why they are categorized as "Crashed" in the *wandb* results. Randomizer seeds are included the sweep again to mitigate variability from model initialization.

The *wandb*'s parallel plot proved insufficient to disentangle the contribution of individual features. The data was thus further explored in enhancement-specific plots. While details are available at `https://api.wandb.ai/links/schups/rqham25y`, key plots are discussed below.

Figure 4.17: Training curves of about 137 runs involved in the ablation study. It can be noticed how the learning rates and schedule are now fixed for all runs. Some runs achieved testing MSE better than .3.

For clarity in enhancement-specific plots, conventions are adopted: (1) green lines represent proposed enhancements, red lines original implementations, (2) y-axes, already logarithmic-scaled, are constrained to reveal end-phase details, and (3) running averages (window: 10) are applied to emphasize trends near rollout increments (w.r.t., curve spikes).

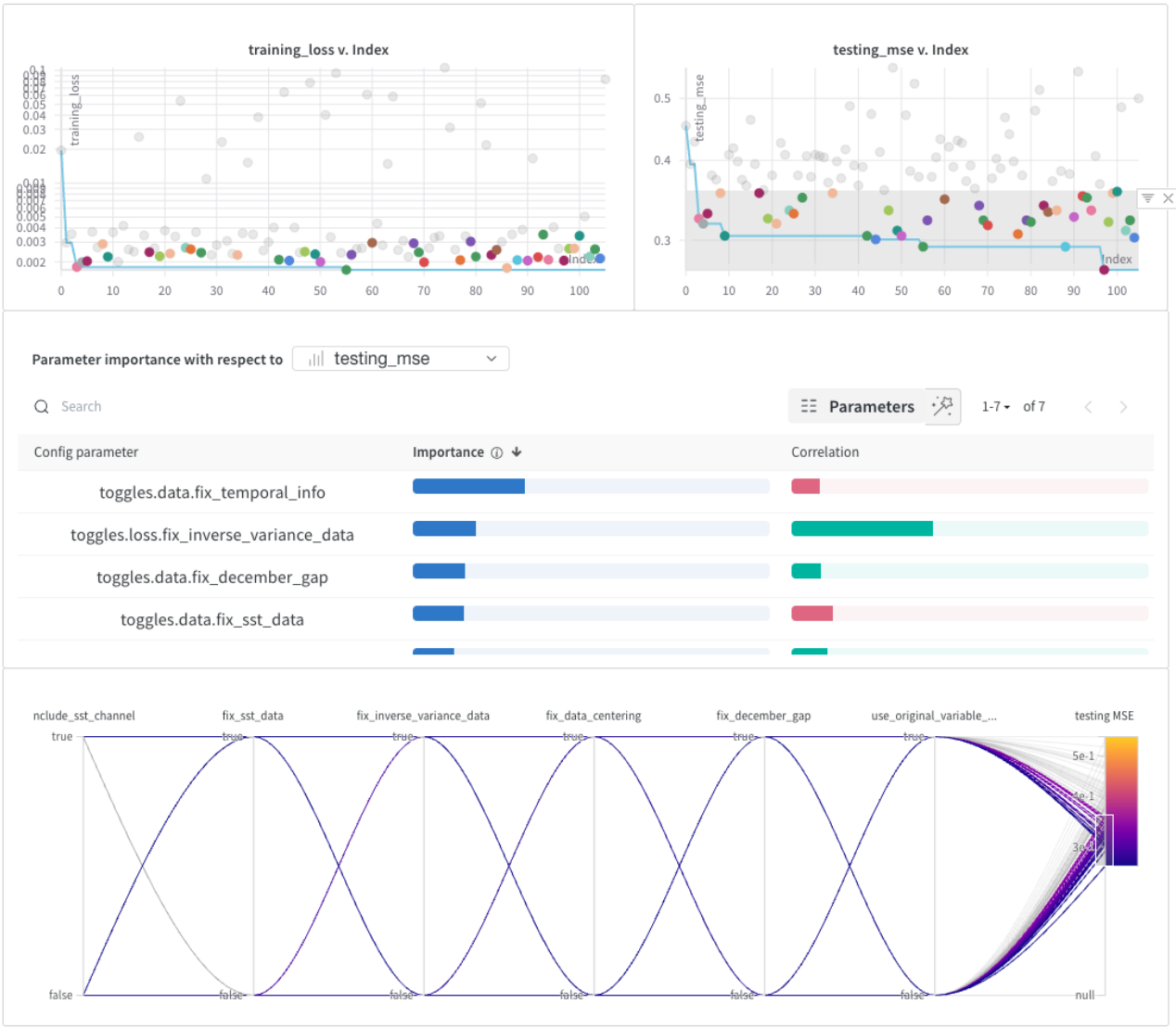


Figure 4.18: The superficial evaluation of importance provided by the default *wandb* interface.

4.7.1 Sea-Surface Temperature Channel

Figure 4.19 illustrates how `sst` data inclusion benefits achievable testing MSE only if the proposed enhancements are also activated.

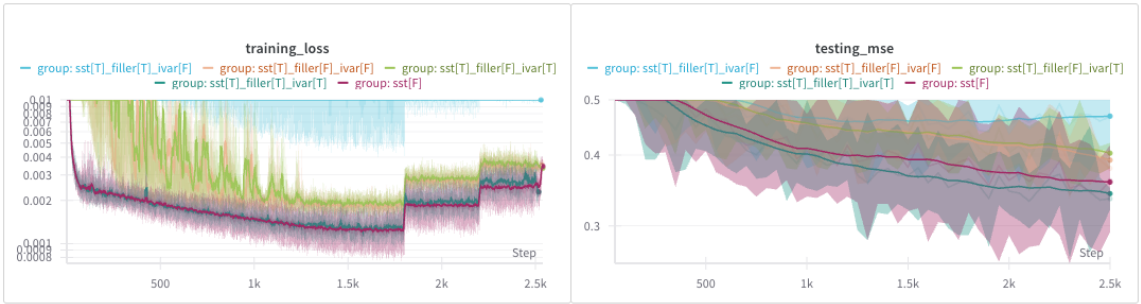


Figure 4.19: Effects on testing MSE from different `sst`-related configurations.

4.7.2 Temporal Information Smoothness Enhancement

Figure 4.20 illustrates how smoother temporal information improves generalization, as shown by consistently lower testing MSE despite similar training losses.

Figure 4.20: Effects on temporal data enhancement on achievable testing MSE.

4.7.3 Misalignment of Top-of-Atmosphere Solar Irradiance

Figure 4.21 shows that, contrary to expectations, the proposed solar irradiance alignment worsens both the loss curve and the generalization. Although values spread is wider for the original implementation, the enhancement's spread is consistently on a higher range.

Figure 4.21: Comparison of training trends for different solar irradiance alignments.

4.7.4 Seasonal Bias Removal

Addressing the December data under-representation reduces improves test MSE trends, highlighting the importance of correcting seasonal discontinuities for better generalization.

Figure 4.22: Comparison of training trends between original and enhancement coverage of later parts of the year.

4.8 FourCastNet as an External Reference

An external reference level is established for comparative purposes by training a FourCastNet model from scratch. This is facilitated by the fact that the FourCastNet codebase is tailored to the dataset used in this study. The codebase, cloned from the official repository, was minimally modified to ensure training data exposure equivalent to that used in the training of GraphCast models against which it will later be compared. Changes are minimal and did not include, e.g., the alignment of loss functions. The model, made up of 74'691'840 parameters, is trained on float32 precision.

Figure 4.23: The training loss of the single FourCastNet model training executed is noticeably smooth. The training loss function is based on MSE.

4.9 Further Experimentation

4.9.1 Model Size Scaling

Following the findings of [40], an increased parameters count was briefly explored. Given the project's tight schedule, a single test was performed.

Aiming at freeing up GPU memory space for the additional parameters, a slightly modified version was developed to disable activations checkpointing (see the `modulus_no_checkpoint` folder and training script). Before hyper-parameters changes, allocated GPU memory dropped from roughly 87 GB to 31, while train iteration runtime went from .28 seconds to .5.

The new model parameters space was obtained by increasing several hyper-parameters: (1) hidden layers dimension from 512 to 1024, (2) hidden layers count from 1 to 2, and (3) processor layers count from 16 to 20 - resulting in trainable parameters count increasing from 35'248'149 to 223'671'316. Allocated GPU memory increased from roughly 31 GB to 73 GB, and the train iteration runtime from .5 seconds to 1.75.

Figure 4.24 illustrates how, while maintaining downward trends, the larger model experiences slower progression. Details available at `https://api.wandb.ai/links/schups/3kfvc9ja`. It is plausible that, given enough training time, the curves converge at levels lower than achieved so far. Nevertheless, due to constraints both on the project execution schedule, but also in the maximum training runtime (recall, the training of this larger model is about 6.5 times slower) this experimental direction is not explored further.

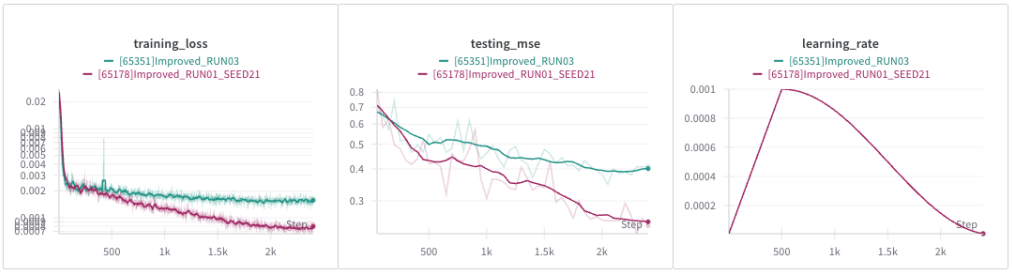


Figure 4.24: Training progression comparison between the model size analysed so far and a significantly bigger variation.

4.9.2 Phase 3 Learning Rate Levels

The training curves obtained so far in the project consistently show the progression of training loss to flatten once rollout length increases above 1. This observation is, given the project constraints, limited to the number of iterations typically involved. Figure 4.25 exemplifies. Although the task of predicting weather over longer periods is supposedly harder, the loss curve is expected to remain on a downward trend. Despite the plateau in loss, the testing MSE values achieve new lows, indicating the effectiveness of the auto-regressive training implementation.

Figure 4.25: The training curves obtained so far and thus unrelated to phase 3 learning rate experimentation. Noticeably, testing MSE reaches new lows on longer rollouts. A visualization tool limitation renders above-range values at the top of the charts.

Given that training is performed at relatively low precision for computational efficiency reasons (i.e., BF16[5]) it is plausible that the learning rate is too low to produce a meaningful magnitude in back-propagation gradients. For this reason, different third-phase learning rates are tested.

[5] https://en.wikipedia.org/wiki/Bfloat16_floating-point_format

Figure 4.26 shows the effects of using different learning rates in the third training phase. Note: (1) the red line is shown as reference, being the best model trained so far (w.r.t., optimized hyperparameters). It increases roll-out every 500 iterations, (2) green lines and blue lines are the subject of this experimentation phase of varying learning rates. The blue lines increase roll-out only once, continuing on two roll-out steps until timeout. Details available at `https://api.wandb.ai/links/schups/o32hbv5e`.

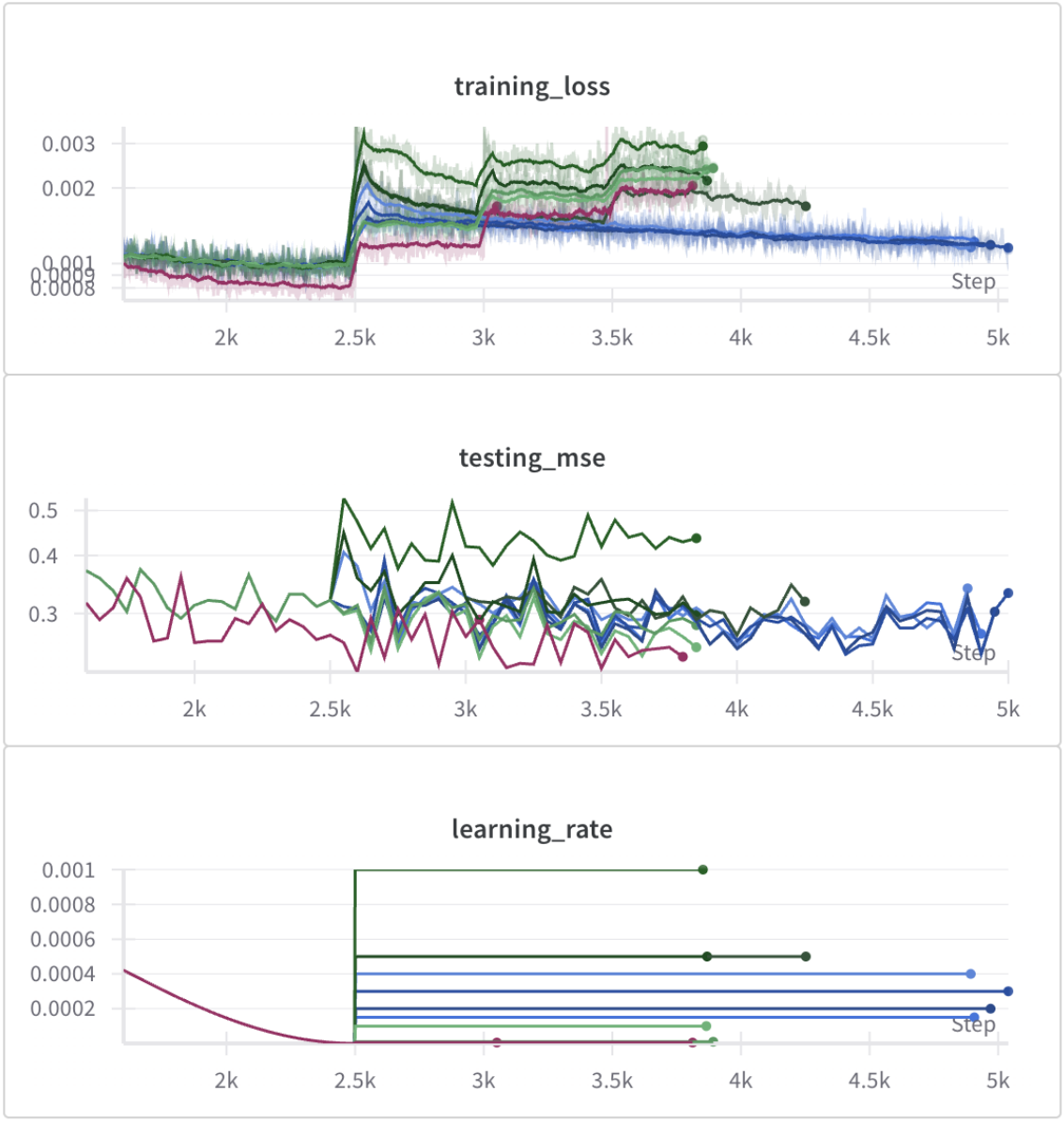


Figure 4.26: Training and testing curves of different third-phase learning rates. The the blue lines are characterized by a single roll-out increase.

While the learning rates values tested produced downward trends in some training loss curves, none improved over previously achieved MSE testing levels. Noticeably, even the blue lines (single roll-out increase) appear to converge on previously achieved levels, at least as far as the runtime limit enables observations. While experimental results fail to reveal new information for clear training improvements, blue lines suggest potential if runtime constraints are lifted and initialization seeds variety increased. The default learning rate, also aligned with the original DeepMind publication, remains the best known option for the current constraints.

4.10 Improved Model Performance

This section presents the best-performing configuration, referred to as `Improved-RUN05`, which integrates all enhancements found to reduce testing MSE, as detailed in earlier sections.

Figure 4.27 contrasts this model with the initial performance baseline established in Section 4.4, highlighting the cumulative effect of proposed enhancements. Details available at `https://api.wandb.ai/links/schups/n2op8xad`.

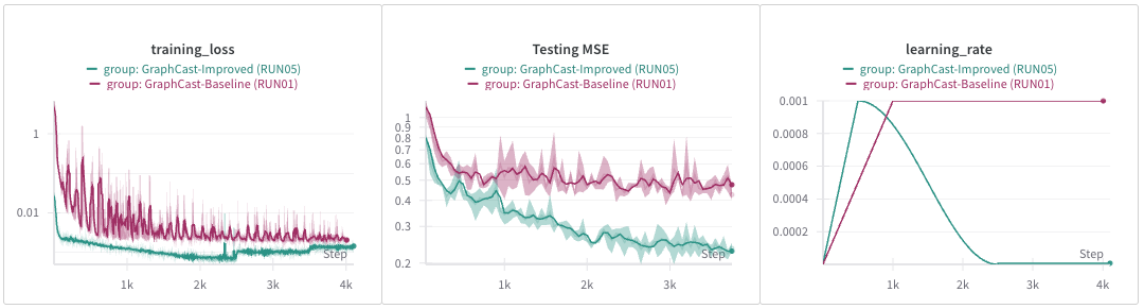


Figure 4.27: Visual comparison of the training curves for the initially established baseline and the aggregated effect of the enhancements proposed in this study.

Figure 4.28 further highlights the individual contributions to testing MSE. The full set of enhancements (`Improved-RUN05`) outperforms both the baseline and partial variants that include either (1) only the improved training schedule or (2) only the proposed enhancements. Shaded regions indicate variance across employed seeds ($n = 3$). The proposed enhancements appear to make a bigger contribution to the improvement, than the adapted training schedule.

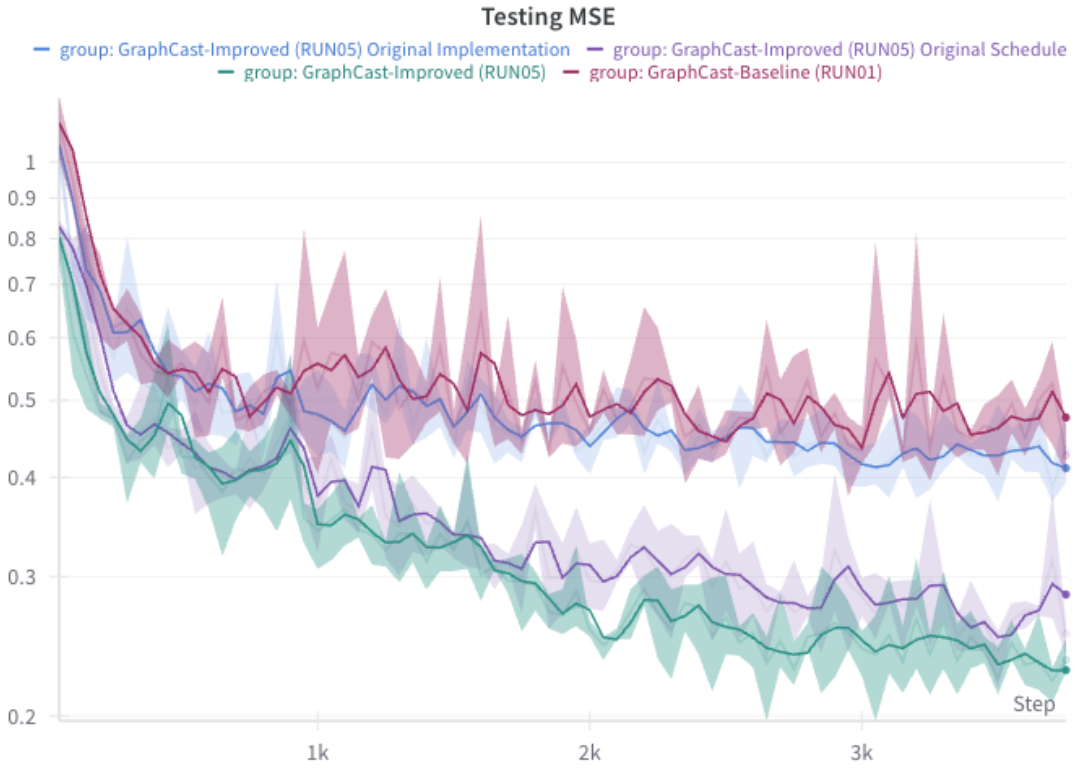


Figure 4.28: Testing MSE comparison across ablated configurations. Shareded regions indicate the spread of values across the trained models (3 models for each line).

4.11 Out-of-Sample Model Performance Results

Whereas previous sections examined training dynamics and in-sample evaluation, this section focuses on out-of-sample generalization, evaluating performance on data not seen in training.

Results are reported using domain-standard metrics: Root Mean Square Error (RMSE) and Anomaly Correlation Coefficient (ACC). Both metrics are presented across aggregated views (e.g., `all` variables at a `global` scale) and individual variable–region combinations. A wider range of combinations is available in Appendix D.

Each plot line displays the aggregated outcome of 27 inference runs, uniformly distributed over the out-of-sample period, evaluated up to 7 days ahead. Three independently trained model instances were used (identical architecture, distinct initializations, and training sequences), except for FourCastNet (blue), where only one model instance instance was trained.

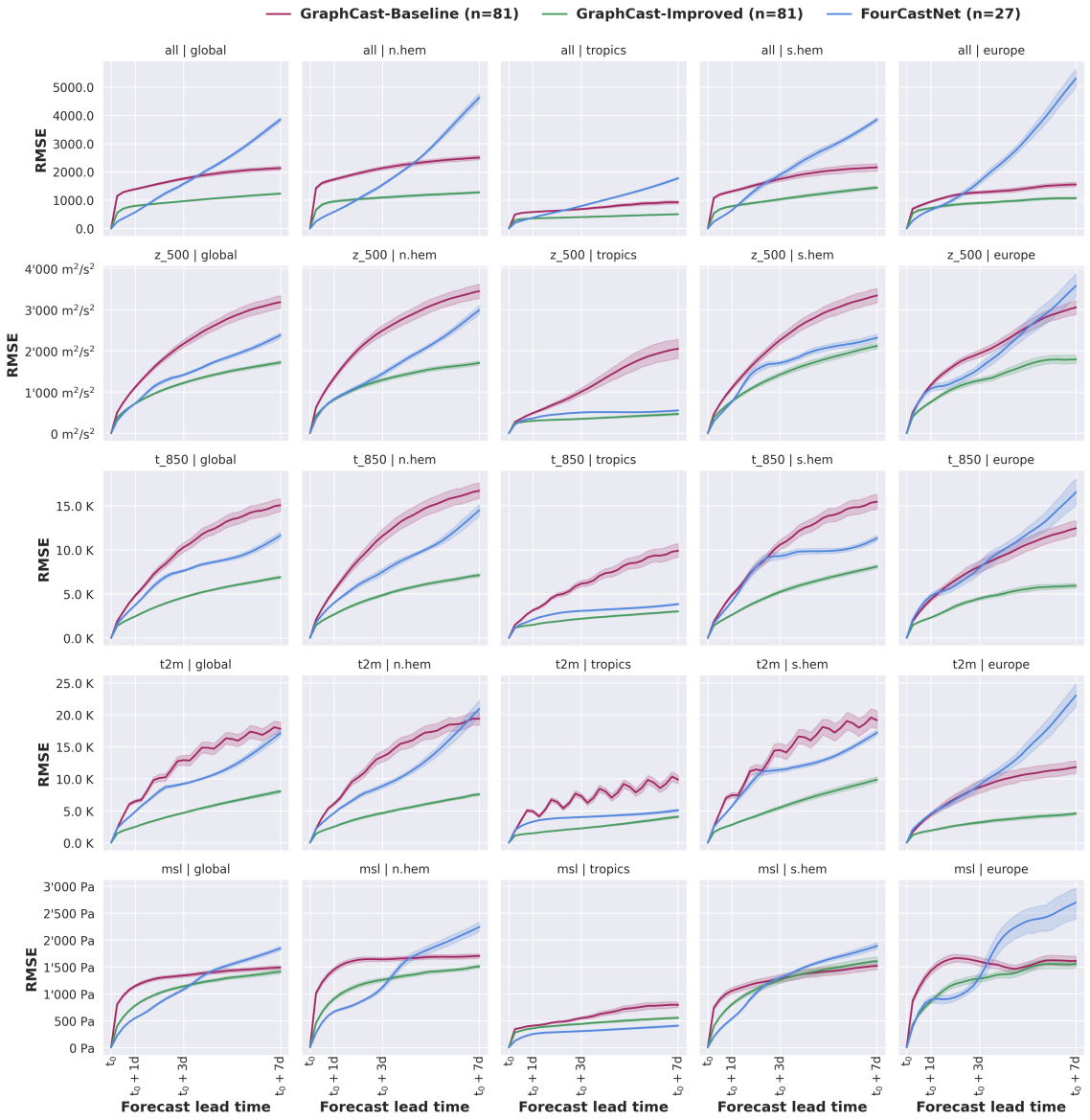


Figure 4.29: Root Mean Square Error (RMSE) across variables and regions. Lower values indicate better forecast skill. Shaded regions reflect variability across inference initializations and model instances.

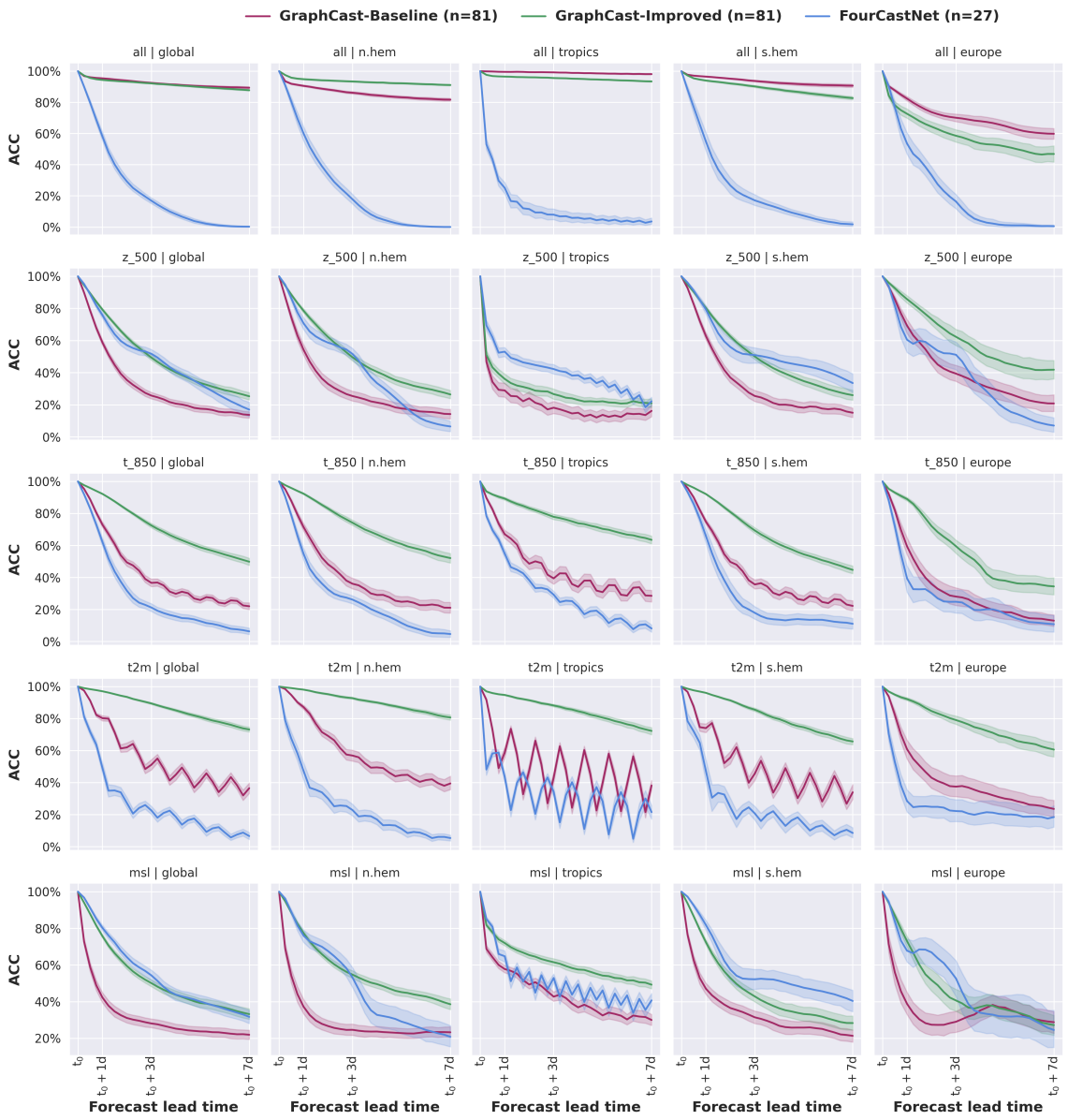


Figure 4.30: Anomaly Correlation Coefficient (ACC) across variables and regions. Higher values indicate better forecast skill. Shaded regions reflect variability across inference initializations and model instances.

Figures 4.29 and 4.30 compare baseline and improved variants of GraphCast across variables (`z_500`, `t_850`, `t2m`, `msl`) and spatial domains (global, hemispheric, and regional). A FourCastNet model trained on equivalent amount of data, is included as a comparative reference representing other established models.

The enhanced GraphCast model, which incorporates the improvements proposed in this study, outperforms the baseline in most cases, particularly in longer lead times and on variables sensitive to diurnal variation. This is especially evident in the `t2m` channel, where the improved model shows greater robustness to intraday patterns.

These findings validate the proposed improvements and highlight their impact on increasing the robustness and long-range forecasting skill of GraphCast.

4.11.1 Quantitative Performance Results

At 1 day lead time.

Variable	Model	global	n.hem	tropics	s.hem	europe
all	GraphCast-Baseline	1392.89 ± 121.98	1736.49 ± 223.59	584.47 ± 38.27	1317.83 ± 192.47	947.60 ± 166.16
all	GraphCast-Improved	801.19 ± 81.45	959.85 ± 165.08	**357.46 ± 26.73**	792.30 ± 140.56	723.46 ± 160.13
all	FourCastNet	**578.41 ± 21.38**	**591.66 ± 29.81**	386.33 ± 14.82	**650.17 ± 49.55**	**656.79 ± 68.39**
z_500	GraphCast-Baseline	1121.78 ± 140.26	1346.72 ± 242.35	492.42 ± 108.32	1104.64 ± 234.28	1169.63 ± 222.00
z_500	GraphCast-Improved	**723.90 ± 71.91**	**817.73 ± 140.66**	**299.95 ± 31.65**	778.96 ± 121.44	**758.07 ± 141.16**
z_500	FourCastNet	725.49 ± 26.94	833.85 ± 59.22	361.50 ± 12.98	**761.75 ± 39.54**	1084.88 ± 210.39
t_850	GraphCast-Baseline	4.82 ± 0.75	5.41 ± 1.18	3.16 ± 0.54	4.88 ± 0.92	4.36 ± 1.10
t_850	GraphCast-Improved	**2.46 ± 0.18**	**2.67 ± 0.41**	**1.49 ± 0.08**	**2.63 ± 0.36**	**2.31 ± 0.35**
t_850	FourCastNet	3.73 ± 0.14	3.98 ± 0.23	2.09 ± 0.06	4.16 ± 0.27	4.76 ± 0.83
t2m	GraphCast-Baseline	6.48 ± 1.27	6.10 ± 1.23	4.90 ± 1.15	7.45 ± 1.88	4.39 ± 1.72
t2m	GraphCast-Improved	**2.48 ± 0.18**	**2.51 ± 0.47**	**1.48 ± 0.10**	**2.81 ± 0.50**	**1.90 ± 0.48**
t2m	FourCastNet	4.87 ± 0.22	4.66 ± 0.24	3.30 ± 0.18	5.73 ± 0.47	4.51 ± 0.54
msl	GraphCast-Baseline	1143.56 ± 140.50	1449.48 ± 212.14	404.96 ± 49.26	1054.81 ± 260.68	1423.36 ± 323.16
msl	GraphCast-Improved	780.59 ± 73.51	907.46 ± 162.66	354.67 ± 35.98	803.31 ± 120.47	**856.94 ± 197.74**
msl	FourCastNet	**547.86 ± 19.87**	**666.46 ± 47.13**	**250.77 ± 25.70**	**536.98 ± 43.63**	889.77 ± 180.92

Table 4.1: RMSE metric scores (mean ± standard deviation) for 1 day forecast lead time. For `GraphCast-Baseline` and `GraphCast-Improved` $n = 81$, and for `FourCastNet` $n = 27$. Lower values are better.

Variable	Model	global	n.hem	tropics	s.hem	europe
all	GraphCast-Baseline	**0.95 ± 0.01**	0.91 ± 0.02	**0.99**	**0.96 ± 0.01**	**0.82 ± 0.06**
all	GraphCast-Improved	0.94 ± 0.01	**0.95 ± 0.02**	0.96 ± 0.01	0.94 ± 0.02	0.73 ± 0.14
all	FourCastNet	0.58 ± 0.07	0.60 ± 0.12	0.25 ± 0.10	0.55 ± 0.16	0.53 ± 0.19
z_500	GraphCast-Baseline	0.59 ± 0.08	0.54 ± 0.16	0.29 ± 0.21	0.63 ± 0.10	0.69 ± 0.20
z_500	GraphCast-Improved	**0.79 ± 0.04**	**0.79 ± 0.06**	0.36 ± 0.15	**0.80 ± 0.05**	**0.86 ± 0.11**
z_500	FourCastNet	0.76 ± 0.05	0.71 ± 0.10	**0.53 ± 0.08**	0.79 ± 0.07	0.60 ± 0.22
t_850	GraphCast-Baseline	0.73 ± 0.07	0.71 ± 0.11	0.67 ± 0.11	0.75 ± 0.07	0.59 ± 0.25
t_850	GraphCast-Improved	**0.92 ± 0.01**	**0.92 ± 0.02**	**0.89 ± 0.04**	**0.92 ± 0.02**	**0.89 ± 0.05**
t_850	FourCastNet	0.62 ± 0.07	0.55 ± 0.10	0.55 ± 0.07	0.66 ± 0.11	0.39 ± 0.21
t2m	GraphCast-Baseline	0.80 ± 0.07	0.87 ± 0.06	0.59 ± 0.14	0.74 ± 0.13	0.61 ± 0.28
t2m	GraphCast-Improved	**0.97 ± 0.01**	**0.98 ± 0.01**	**0.95 ± 0.02**	**0.96 ± 0.01**	**0.92 ± 0.04**
t2m	FourCastNet	0.49 ± 0.09	0.47 ± 0.14	0.43 ± 0.08	0.47 ± 0.18	0.28 ± 0.19
msl	GraphCast-Baseline	0.42 ± 0.13	0.37 ± 0.18	0.58 ± 0.09	0.47 ± 0.14	0.39 ± 0.28
msl	GraphCast-Improved	0.76 ± 0.04	**0.77 ± 0.05**	**0.72 ± 0.07**	0.72 ± 0.06	**0.73 ± 0.14**
msl	FourCastNet	**0.80 ± 0.04**	0.76 ± 0.09	0.65 ± 0.08	**0.82 ± 0.07**	0.68 ± 0.18

Table 4.2: ACC metric scores (mean ± standard deviation) for 1 day forecast lead time. For `GraphCast-Baseline` and `GraphCast-Improved` $n = 81$, and for `FourCastNet` $n = 27$. Higher values are better.

At 3 days lead time.

Variable	Model	global	n.hem	tropics	s.hem	europe
all	GraphCast-Baseline	1763.36 ± 217.31	2133.14 ± 295.54	688.35 ± 96.44	1751.01 ± 382.69	1278.67 ± 235.86
all	GraphCast-Improved	**968.35 ± 86.19**	**1097.43 ± 161.01**	**406.10 ± 42.66**	**1036.54 ± 187.04**	**906.27 ± 175.71**
all	FourCastNet	1581.82 ± 110.69	1549.17 ± 119.10	798.19 ± 33.61	1903.62 ± 254.61	1648.57 ± 327.76
z_500	GraphCast-Baseline	2171.75 ± 401.09	2492.44 ± 554.35	1009.14 ± 381.87	2265.88 ± 497.11	1933.33 ± 400.83
z_500	GraphCast-Improved	**1222.12 ± 130.34**	**1294.07 ± 195.10**	**350.52 ± 53.71**	**1425.09 ± 281.68**	**1280.59 ± 297.41**
z_500	FourCastNet	1417.00 ± 84.03	1444.09 ± 160.14	505.44 ± 25.56	1704.11 ± 145.02	1567.73 ± 290.42
t_850	GraphCast-Baseline	10.27 ± 2.46	11.55 ± 3.45	6.16 ± 1.67	10.58 ± 2.56	8.07 ± 3.08
t_850	GraphCast-Improved	**4.60 ± 0.40**	**4.85 ± 0.70**	**2.18 ± 0.17**	**5.23 ± 0.93**	**4.46 ± 0.90**
t_850	FourCastNet	7.62 ± 0.41	7.49 ± 1.01	3.07 ± 0.12	9.28 ± 0.97	7.89 ± 2.00
t2m	GraphCast-Baseline	12.91 ± 3.62	13.46 ± 4.13	7.27 ± 1.91	14.47 ± 4.77	8.65 ± 4.44
t2m	GraphCast-Improved	**4.67 ± 0.55**	**4.63 ± 0.83**	**2.25 ± 0.28**	**5.50 ± 1.28**	**3.17 ± 0.96**
t2m	FourCastNet	9.22 ± 0.48	8.85 ± 0.98	4.01 ± 0.23	11.39 ± 0.98	8.97 ± 1.75
msl	GraphCast-Baseline	1336.49 ± 143.56	1637.23 ± 183.71	546.59 ± 116.55	1283.75 ± 330.03	1598.42 ± 356.49
msl	GraphCast-Improved	1136.74 ± 108.35	1255.85 ± 168.50	437.21 ± 60.01	**1252.25 ± 268.84**	**1281.14 ± 282.48**
msl	FourCastNet	**1079.62 ± 68.92**	**1116.42 ± 124.50**	**303.46 ± 23.91**	1297.39 ± 109.19	1292.12 ± 336.68

Table 4.3: RMSE metric scores (mean ± standard deviation) for 3 days forecast lead time. For `GraphCast-Baseline` and `GraphCast-Improved` $n = 81$, and for `FourCastNet` $n = 27$. Lower values are better.

Variable	Model	global	n.hem	tropics	s.hem	europe
all	GraphCast-Baseline	**0.92 ± 0.02**	0.86 ± 0.03	**0.99 ± 0.01**	**0.94 ± 0.03**	**0.70 ± 0.11**
all	GraphCast-Improved	0.92 ± 0.01	**0.93 ± 0.02**	0.96 ± 0.01	0.90 ± 0.03	0.59 ± 0.18
all	FourCastNet	0.17 ± 0.05	0.18 ± 0.08	0.08 ± 0.09	0.17 ± 0.09	0.16 ± 0.12
z_500	GraphCast-Baseline	0.26 ± 0.09	0.25 ± 0.14	0.18 ± 0.18	0.25 ± 0.15	0.39 ± 0.24
z_500	GraphCast-Improved	0.49 ± 0.07	0.50 ± 0.11	0.27 ± 0.14	0.50 ± 0.09	**0.62 ± 0.25**
z_500	FourCastNet	**0.52 ± 0.07**	**0.52 ± 0.10**	**0.42 ± 0.08**	**0.51 ± 0.13**	0.51 ± 0.24
t_850	GraphCast-Baseline	0.37 ± 0.11	0.36 ± 0.15	0.39 ± 0.17	0.36 ± 0.13	0.28 ± 0.25
t_850	GraphCast-Improved	**0.72 ± 0.06**	**0.74 ± 0.07**	**0.78 ± 0.08**	**0.69 ± 0.08**	**0.59 ± 0.19**
t_850	FourCastNet	0.21 ± 0.06	0.26 ± 0.07	0.29 ± 0.06	0.18 ± 0.11	0.25 ± 0.19
t2m	GraphCast-Baseline	0.51 ± 0.15	0.57 ± 0.16	0.43 ± 0.16	0.46 ± 0.20	0.37 ± 0.31
t2m	GraphCast-Improved	**0.89 ± 0.03**	**0.93 ± 0.03**	**0.88 ± 0.04**	**0.85 ± 0.05**	**0.79 ± 0.12**
t2m	FourCastNet	0.23 ± 0.07	0.23 ± 0.10	0.34 ± 0.08	0.21 ± 0.12	0.22 ± 0.19
msl	GraphCast-Baseline	0.28 ± 0.11	0.25 ± 0.14	0.43 ± 0.12	0.31 ± 0.15	0.29 ± 0.28
msl	GraphCast-Improved	0.50 ± 0.09	**0.55 ± 0.09**	**0.61 ± 0.09**	0.43 ± 0.14	0.41 ± 0.27
msl	FourCastNet	**0.54 ± 0.08**	0.53 ± 0.14	0.53 ± 0.07	**0.52 ± 0.15**	**0.53 ± 0.26**

Table 4.4: ACC metric scores (mean ± standard deviation) for 3 days forecast lead time. For `GraphCast-Baseline` and `GraphCast-Improved` $n = 81$, and for `FourCastNet` $n = 27$. Higher values are better.

At a week lead time.

Variable	Model	global	n.hem	tropics	s.hem	europe
all	GraphCast-Baseline	2133.43 ± 322.02	2503.11 ± 363.07	927.46 ± 265.62	2160.69 ± 625.49	1556.20 ± 355.93
all	GraphCast-Improved	**1234.47 ± 121.65**	**1279.90 ± 159.22**	**506.86 ± 71.98**	**1444.74 ± 249.84**	**1073.33 ± 190.08**
all	FourCastNet	3846.41 ± 212.51	4621.02 ± 393.23	1777.57 ± 102.43	3847.10 ± 207.76	5298.84 ± 900.89
z_500	GraphCast-Baseline	3181.93 ± 706.21	3447.51 ± 786.50	2050.72 ± 1063.91	3340.44 ± 809.85	3054.22 ± 802.87
z_500	GraphCast-Improved	**1718.22 ± 200.75**	**1706.17 ± 239.15**	**469.51 ± 126.19**	**2117.33 ± 341.81**	**1792.40 ± 464.07**
z_500	FourCastNet	2377.16 ± 178.13	2988.43 ± 280.57	555.54 ± 39.71	2319.01 ± 223.07	3574.71 ± 784.75
t_850	GraphCast-Baseline	15.03 ± 3.44	16.66 ± 4.21	9.88 ± 3.52	15.43 ± 3.90	12.43 ± 3.81
t_850	GraphCast-Improved	**6.88 ± 0.72**	**7.11 ± 0.91**	**3.02 ± 0.39**	**8.09 ± 1.14**	**5.91 ± 1.23**
t_850	FourCastNet	11.61 ± 1.05	14.47 ± 1.87	3.84 ± 0.28	11.29 ± 0.87	16.51 ± 3.89
t2m	GraphCast-Baseline	17.80 ± 4.65	19.35 ± 5.03	9.85 ± 3.22	19.15 ± 6.65	11.81 ± 4.48
t2m	GraphCast-Improved	**8.03 ± 1.23**	**7.56 ± 1.02**	**4.07 ± 1.01**	**9.85 ± 2.09**	**4.56 ± 1.14**
t2m	FourCastNet	17.11 ± 1.68	20.92 ± 3.31	5.08 ± 0.56	17.19 ± 1.18	22.99 ± 4.87
msl	GraphCast-Baseline	1486.42 ± 164.36	1701.33 ± 192.62	791.68 ± 249.53	**1516.69 ± 363.14**	1608.44 ± 386.56
msl	GraphCast-Improved	**1408.20 ± 178.85**	**1505.78 ± 166.12**	552.12 ± 89.09	1605.14 ± 368.45	**1555.21 ± 344.63**
msl	FourCastNet	1840.68 ± 107.67	2237.80 ± 230.43	**404.04 ± 44.08**	1886.55 ± 172.54	2693.36 ± 771.28

Table 4.5: RMSE metric scores (mean ± standard deviation) for 7 days forecast lead time. For `GraphCast-Baseline` and `GraphCast-Improved` $n = 81$, and for `FourCastNet` $n = 27$. Lower values are better.

Variable	Model	global	n.hem	tropics	s.hem	europe
all	GraphCast-Baseline	**0.89 ± 0.03**	0.82 ± 0.04	**0.98 ± 0.01**	**0.91 ± 0.05**	**0.60 ± 0.17**
all	GraphCast-Improved	0.88 ± 0.02	**0.91 ± 0.02**	0.93 ± 0.02	0.83 ± 0.05	0.47 ± 0.23
all	FourCastNet	0.00 ± 0.01	0.00	0.04 ± 0.05	0.02 ± 0.04	0.01 ± 0.02
z_500	GraphCast-Baseline	0.14 ± 0.09	0.14 ± 0.13	0.16 ± 0.17	0.15 ± 0.13	0.21 ± 0.24
z_500	GraphCast-Improved	**0.25 ± 0.11**	**0.26 ± 0.12**	0.21 ± 0.11	0.26 ± 0.15	**0.42 ± 0.27**
z_500	FourCastNet	0.17 ± 0.12	0.06 ± 0.09	**0.22 ± 0.07**	**0.33 ± 0.16**	0.07 ± 0.12
t_850	GraphCast-Baseline	0.22 ± 0.09	0.21 ± 0.15	0.29 ± 0.16	0.22 ± 0.14	0.13 ± 0.16
t_850	GraphCast-Improved	**0.50 ± 0.10**	**0.52 ± 0.14**	**0.63 ± 0.11**	**0.45 ± 0.11**	**0.34 ± 0.24**
t_850	FourCastNet	0.06 ± 0.05	0.05 ± 0.06	0.08 ± 0.06	0.11 ± 0.09	0.11 ± 0.14
t2m	GraphCast-Baseline	0.36 ± 0.16	0.39 ± 0.18	0.38 ± 0.14	0.34 ± 0.21	0.23 ± 0.24
t2m	GraphCast-Improved	**0.73 ± 0.07**	**0.81 ± 0.07**	**0.72 ± 0.10**	**0.66 ± 0.09**	**0.61 ± 0.20**
t2m	FourCastNet	0.07 ± 0.06	0.05 ± 0.05	0.21 ± 0.11	0.08 ± 0.08	0.18 ± 0.18
msl	GraphCast-Baseline	0.22 ± 0.12	0.23 ± 0.13	0.30 ± 0.13	0.21 ± 0.16	**0.29 ± 0.28**
msl	GraphCast-Improved	**0.33 ± 0.13**	**0.39 ± 0.13**	**0.49 ± 0.10**	0.28 ± 0.19	0.27 ± 0.27
msl	FourCastNet	0.32 ± 0.10	0.21 ± 0.16	0.41 ± 0.08	**0.40 ± 0.16**	0.24 ± 0.27

Table 4.6: ACC metric scores (mean ± standard deviation) for 7 days forecast lead time. For `GraphCast-Baseline` and `GraphCast-Improved` $n = 81$, and for `FourCastNet` $n = 27$. Higher values are better.

4.11.2 Inference Artifacts Samples

For illustrative purposes, this section includes inference artifacts examples for the `z_500`, `t2m`, `t_850`, and `tcwv` channels, comparing predictive performances between `GraphCast-Baseline-RUN01` and `GraphCast-Improved-RUN05`.

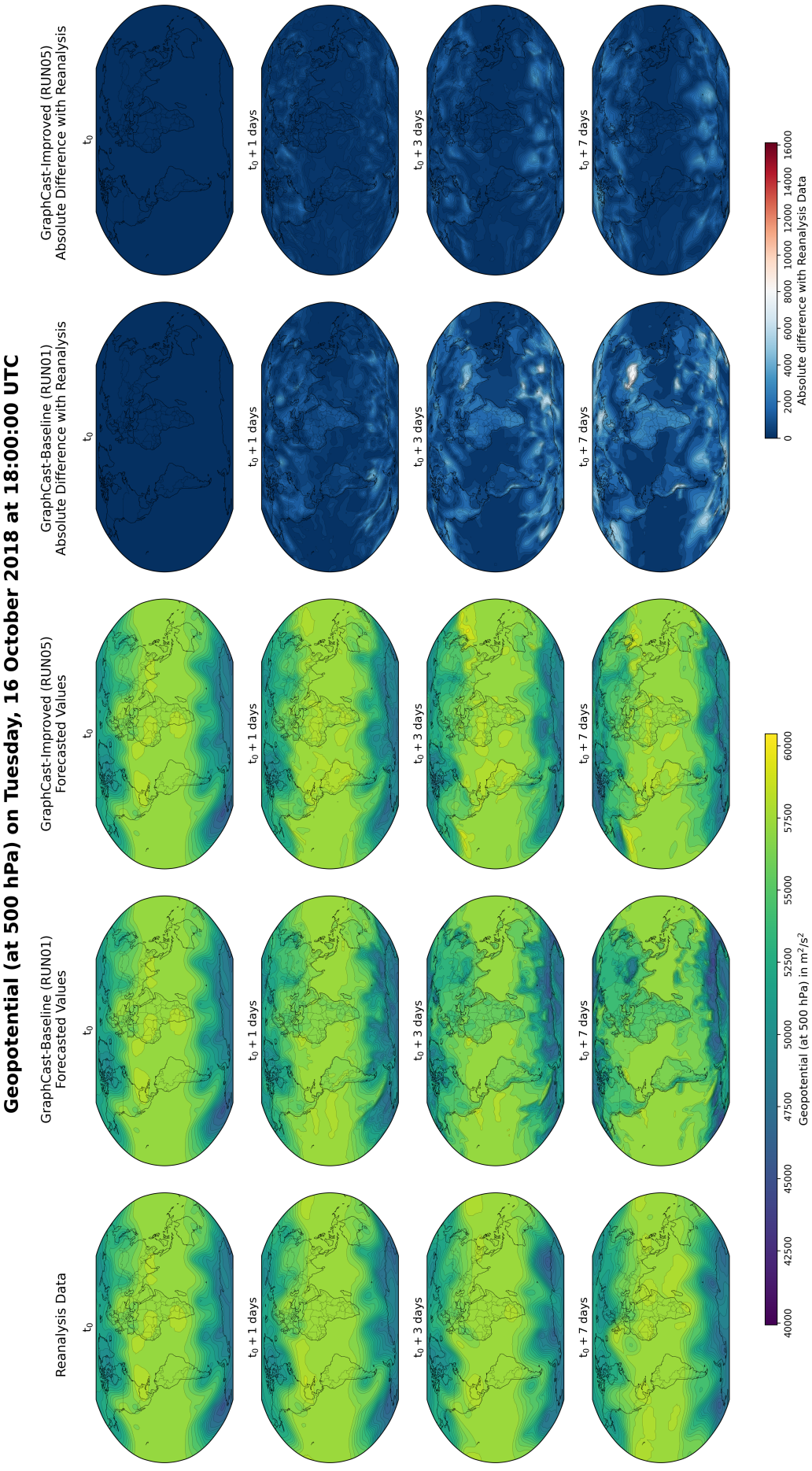


Figure 4.31: Forecasts comparison for the Geopotential channel (at 500 hPa). Inference was initialized on Tuesday, 16 October 2018 at 18:00:00 UTC. Models `GraphCast-Baseline-RUN01` and `GraphCast-Improved-RUN05` both trained on seed 21.

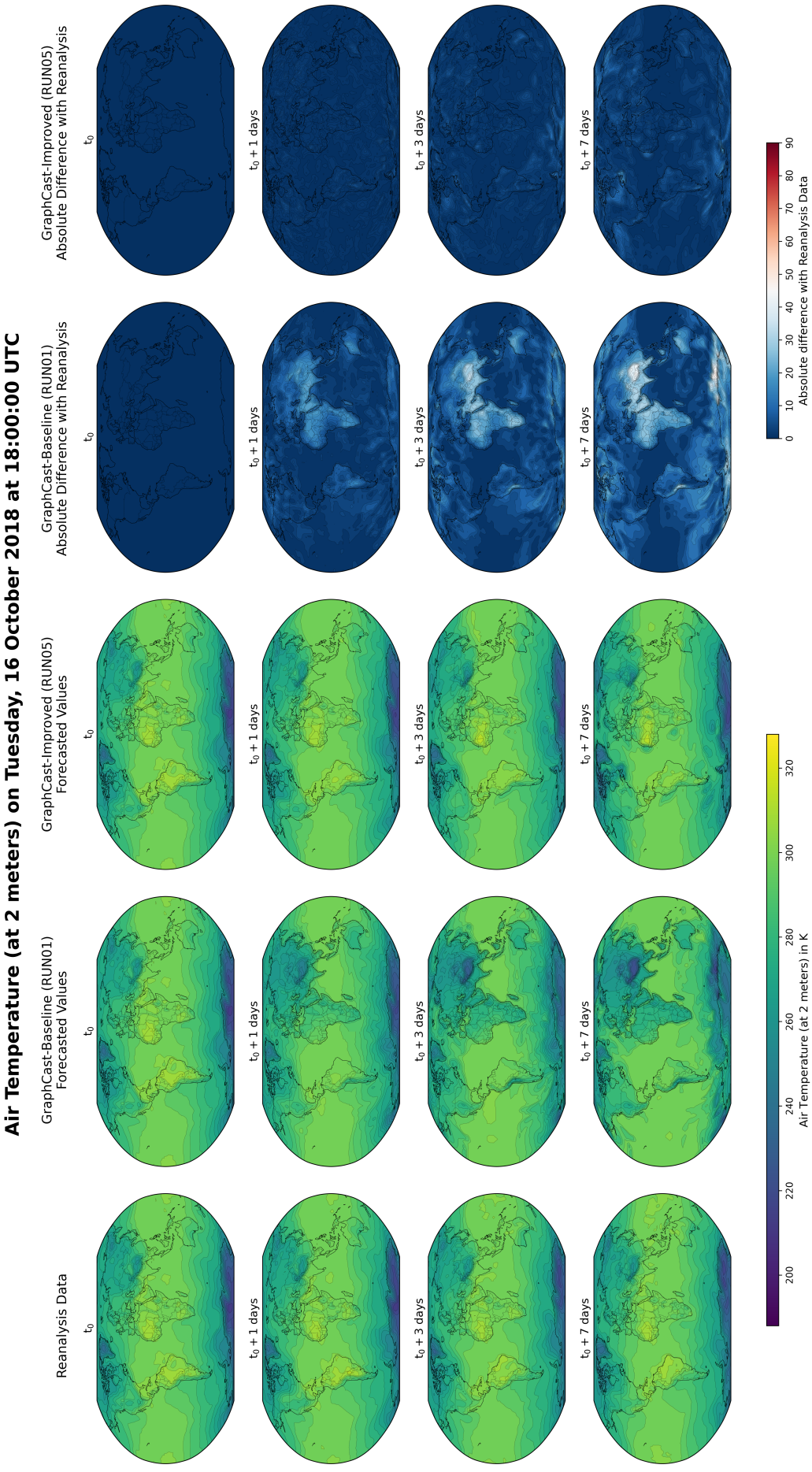


Figure 4.32: Forecasts comparison for the near-surface Temperature channel. Inference was initialized on Tuesday, 16 October 2018 at 18:00:00 UTC. Models `GraphCast-Baseline-RUN01` and `GraphCast-Improved-RUN05` both trained with seed 21.

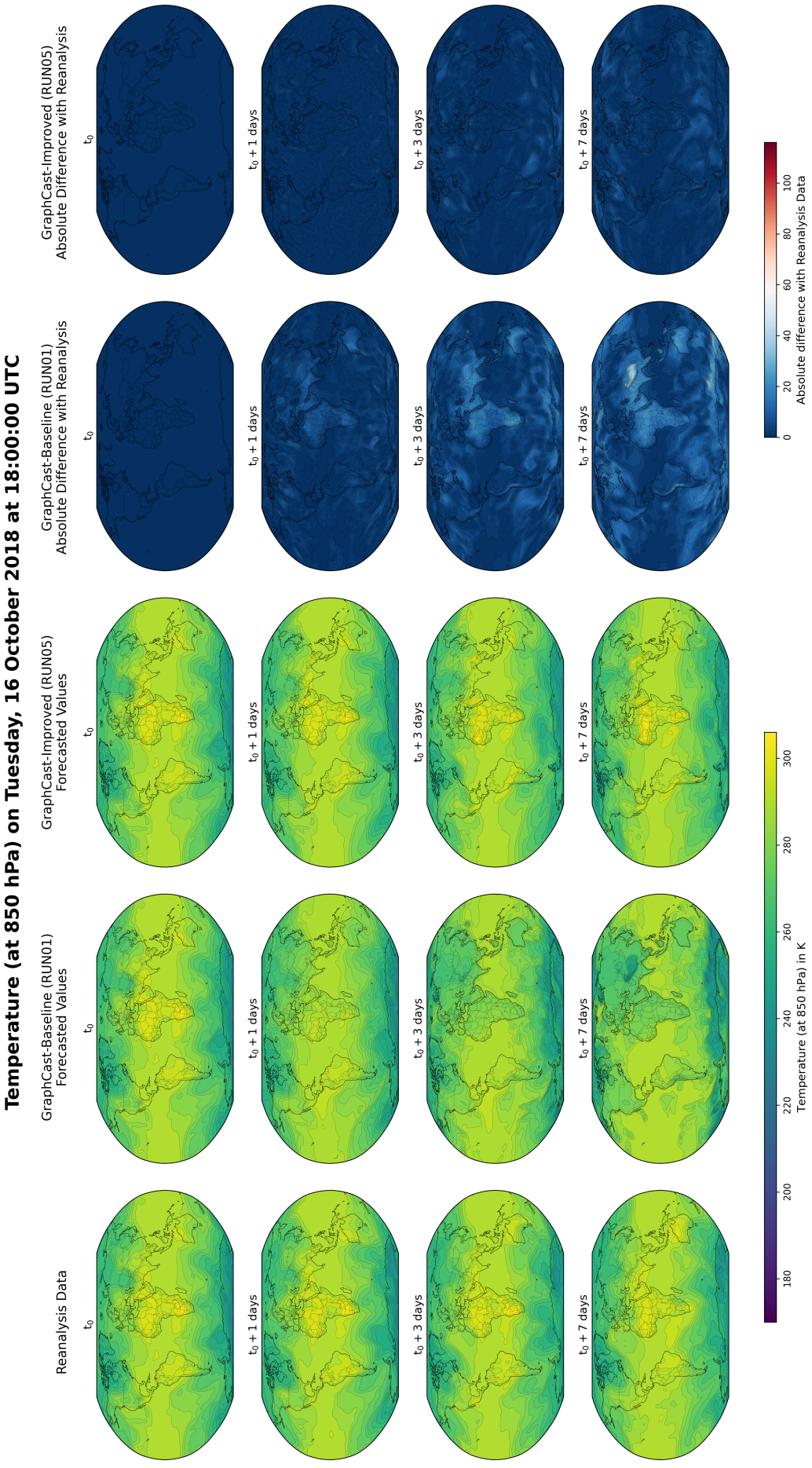


Figure 4.33: Forecasts comparison for the Temperature channel (at 850 hPa). Inference was initialized on Tuesday, 16 October 2018 at 18:00:00 UTC. Models `GraphCast-Baseline-RUN01` and `GraphCast-Improved-RUN05` both trained with seed 21.

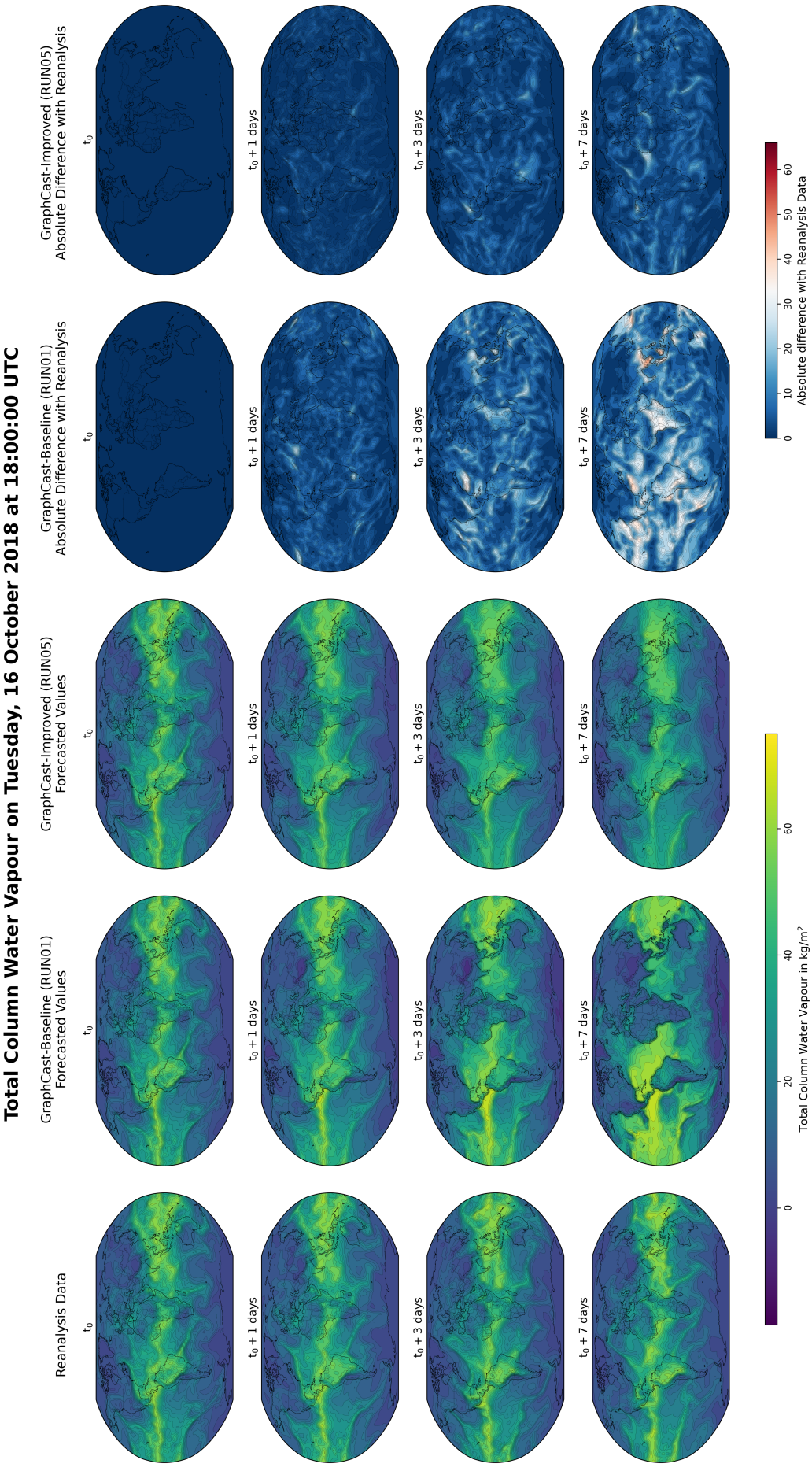


Figure 4.34: Forecasts comparison for the Total Column Water Vapour channel. Inference was initialized on Tuesday, 16 October 2018 at 18:00:00 UTC. Models `GraphCast-Baseline-RUN01` and `GraphCast-Improved-RUN05` both trained with seed 21.

4.12 Statistical Significance Tests

To assess performance differences between the baseline and improved GraphCast models, pairwise statistical comparisons are conducted using matched seeds and initial conditions, ensuring equivalence in experimental setup. As described in Section 3.3, the non-parametric Wilcoxon signed-rank test is employed to evaluate whether differences in model performance are statistically significant.

The null hypothesis states that there is no difference in performance between the two models result distributions. This hypothesis is rejected at a standard significance level of $\alpha = 0.05$. For comparisons where the null hypothesis is rejected, the Cohen's d is also reported to quantify the magnitude of the effect, providing additional insight into the practical relevance of the observed differences.

Metric	Variable	Lead time	global	n.hem	tropics	s.hem	europe
RMSE	all	1 day	**5.34e-15** (**d=5.6**)	**5.35e-15** (**d=4.3**)	**5.34e-15** (**d=5.7**)	**5.35e-15** (**d=4.1**)	**6.45e-15** (**d=1.8**)
RMSE	all	3 days	**5.35e-15** (**d=3.5**)	**5.36e-15** (**d=3.1**)	**5.35e-15** (**d=2.9**)	**5.56e-15** (**d=2.0**)	**1.09e-14** (**d=1.3**)
RMSE	all	7 days	**5.34e-15** (**d=2.7**)	**5.36e-15** (**d=2.9**)	**5.36e-15** (**d=1.8**)	**4.29e-13** (**d=1.2**)	**5.81e-14** (**d=1.2**)
RMSE	z_500	1 day	**5.34e-15** (**d=3.1**)	**5.55e-15** (**d=2.3**)	**5.35e-15** (**d=1.8**)	**5.35e-15** (**d=1.8**)	**6.95e-15** (**d=1.7**)
RMSE	z_500	3 days	**5.35e-15** (**d=2.4**)	**5.36e-15** (**d=1.9**)	**5.36e-15** (**d=1.7**)	**5.35e-15** (**d=2.0**)	**6.36e-14** (**d=1.4**)
RMSE	z_500	7 days	**5.35e-15** (**d=2.1**)	**5.36e-15** (**d=1.9**)	**5.78e-15** (**d=1.5**)	**1.49e-14** (**d=1.5**)	**1.09e-14** (**d=1.5**)
RMSE	t_850	1 day	**5.35e-15** (**d=3.1**)	**5.36e-15** (**d=2.4**)	**5.36e-15** (**d=3.0**)	**5.35e-15** (**d=2.8**)	**5.56e-15** (**d=1.9**)
RMSE	t_850	3 days	**5.36e-15** (**d=2.4**)	**5.36e-15** (**d=1.9**)	**5.36e-15** (**d=2.4**)	**5.35e-15** (**d=2.5**)	**7.89e-14** (**d=1.1**)
RMSE	t_850	7 days	**5.36e-15** (**d=2.5**)	**5.36e-15** (**d=2.4**)	**5.36e-15** (**d=2.0**)	**5.36e-15** (**d=2.0**)	**5.36e-15** (**d=1.7**)
RMSE	t2m	1 day	**5.36e-15** (**d=3.3**)	**5.36e-15** (**d=2.9**)	**5.36e-15** (**d=3.0**)	**5.36e-15** (**d=2.7**)	**5.36e-15** (**d=1.4**)
RMSE	t2m	3 days	**5.36e-15** (**d=2.4**)	**5.36e-15** (**d=2.1**)	**5.36e-15** (**d=2.6**)	**5.36e-15** (**d=2.0**)	**6.83e-15** (**d=1.2**)
RMSE	t2m	7 days	**5.36e-15** (**d=2.3**)	**5.36e-15** (**d=2.5**)	**5.36e-15** (**d=1.8**)	**6.34e-15** (**d=1.5**)	**9.37e-15** (**d=1.5**)
RMSE	msl	1 day	**5.34e-15** (**d=3.4**)	**5.35e-15** (**d=3.8**)	**4.72e-13** (**d=1.2**)	**5.55e-15** (**d=1.4**)	**5.56e-15** (**d=2.2**)
RMSE	msl	3 days	**1.91e-14** (**d=1.7**)	**5.33e-15** (**d=2.8**)	**4.54e-12** (**d=1.0**)	3.61e-01	**4.87e-12** (**d=1.0**)
RMSE	msl	7 days	**1.25e-04** (**d=0.5**)	**2.00e-12** (**d=1.1**)	**2.15e-11** (**d=1.0**)	**4.05e-02** (d=-0.3)	1.81e-01
ACC	all	1 day	**7.04e-11** (d=1.0)	**1.10e-14** (**d=-2.2**)	**4.35e-15** (d=4.2)	**8.44e-15** (d=1.9)	**2.09e-12** (d=0.9)
ACC	all	3 days	6.66e-02	**5.30e-15** (**d=-1.9**)	**4.84e-15** (d=3.2)	**3.49e-11** (d=1.0)	**4.78e-08** (d=0.7)
ACC	all	7 days	**6.56e-05** (d=0.5)	**5.52e-15** (**d=-1.8**)	**5.11e-15** (d=2.8)	**4.07e-13** (d=1.4)	**3.40e-06** (d=0.5)
ACC	z_500	1 day	**5.34e-15** (**d=-2.8**)	**5.55e-15** (**d=-1.7**)	**1.73e-03** (**d=-0.4**)	**5.34e-15** (**d=-2.1**)	**1.69e-14** (**d=-1.3**)
ACC	z_500	3 days	**5.36e-15** (**d=-2.5**)	**2.53e-14** (**d=-1.6**)	**2.46e-04** (**d=-0.5**)	**2.72e-14** (**d=-1.6**)	**6.31e-10** (**d=-0.9**)
ACC	z_500	7 days	**5.62e-10** (**d=-0.9**)	**3.77e-08** (**d=-0.7**)	**1.10e-02** (**d=-0.3**)	**1.01e-06** (**d=-0.6**)	**1.82e-06** (**d=-0.6**)
ACC	t_850	1 day	**5.31e-15** (**d=-2.6**)	**5.34e-15** (**d=-2.0**)	**5.35e-15** (**d=-2.3**)	**5.31e-15** (**d=-2.4**)	**5.34e-15** (**d=-1.3**)
ACC	t_850	3 days	**5.35e-15** (**d=-4.0**)	**5.36e-15** (**d=-2.7**)	**5.35e-15** (**d=-2.1**)	**5.35e-15** (**d=-2.5**)	**1.33e-12** (**d=-1.1**)
ACC	t_850	7 days	**5.36e-15** (**d=-2.6**)	**1.49e-14** (**d=-1.7**)	**8.38e-15** (**d=-1.6**)	**1.28e-13** (**d=-1.4**)	**2.45e-09** (**d=-0.8**)
ACC	t2m	1 day	**5.33e-15** (**d=-2.4**)	**5.31e-15** (**d=-1.9**)	**5.35e-15** (**d=-2.6**)	**5.34e-15** (**d=-1.8**)	**7.83e-15** (**d=-1.1**)
ACC	t2m	3 days	**5.35e-15** (**d=-2.7**)	**5.35e-15** (**d=-2.2**)	**5.36e-15** (**d=-2.8**)	**5.36e-15** (**d=-2.2**)	**2.45e-13** (**d=-1.2**)
ACC	t2m	7 days	**5.36e-15** (**d=-2.4**)	**5.99e-15** (**d=-2.1**)	**5.36e-15** (**d=-2.5**)	**9.84e-15** (**d=-1.7**)	**4.88e-12** (**d=-1.2**)
ACC	msl	1 day	**5.34e-15** (**d=-3.1**)	**5.35e-15** (**d=-2.7**)	**5.54e-15** (**d=-2.1**)	**5.35e-15** (**d=-1.9**)	**6.22e-15** (**d=-1.6**)
ACC	msl	3 days	**5.35e-15** (**d=-2.4**)	**5.36e-15** (**d=-2.8**)	**2.16e-14** (**d=-1.7**)	**3.02e-09** (**d=-0.8**)	**1.54e-06** (**d=-0.6**)
ACC	msl	7 days	**1.24e-11** (**d=-1.1**)	**9.95e-14** (**d=-1.4**)	**1.32e-13** (**d=-1.5**)	**1.25e-04** (**d=-0.4**)	9.61e-01

Table 4.7: Wilcoxon signed-rank test p-values for pairwise comparison between the baseline and improved GraphCast models. Bold values indicate statistically significant differences ($p < 0.05$). The corresponding effect size (Cohen's d) is reported to quantify the magnitude of the difference. Bold values indicate that the improved model had better values for the metric.

5. Discussion

Earlier chapters highlighted the significant potential for enhancements present in the Modulus re-implementation of GraphCast.

Refactoring the codebase and optimizing the training schedule improved efficiency from approximately 900 to 4,000 iterations within the same 30-minute runtime constraint. This enhancement was essential for evaluating the impact of proposed model improvements, as certain benefits only emerged after a sufficient number of iterations. Without this efficiency gain, these effects may have remained undetectable. While replicating DeepMind's full training schedule (around 300,000 iterations) was infeasible due to the tight compute budget, the increased efficiency enabled a longer exploration of it, providing a stronger foundation for extrapolating potential performance gains to an hypothetical full-training campaign.

The baseline model, based on the original Modulus implementation, achieved reasonable accuracy at short lead times but showed rapid deteriorated beyond the first forecast day. In contrast, the proposed enhancements, including (1) improved preparation of sea surface temperature (`sst`) data, (2) smoother additive temporal signals, (3) corrected geographical alignment of real-world physical phenomena in the data, and (4) elimination of seasonal-representation bias in training data, jointly delivered substantial improvements. These gains were generally consistent across variables, lead times, and regions, enhancing both absolute performance metrics and robustness, particularly in modeling diurnal effects.

Although the enhanced GraphCast model did not universally outperform the baseline in all cases, its results were superior in the vast majority of cases analysed. In most metrics, its performance also exceeded that of a FourCastNet model trained on the same amount of data, despite the latter having twice as many parameters and operating on higher precision (`float32`). However, for the channels where it fails to achieve the best values, further analysis is necessary, and together with a more exhaustive exploration of architectural parameters (e.g., message-passing depth, graph size), these remain for future work because of project time limitations.

An ablation study revealed that the inclusion and improved preprocessing of `sst` data contributed most significantly to the observed improvements. This suggests that the original FourCastNet publication might have benefited similarly from these measures. Notably, while the `sst` channel was present in their dataset and in their normalization statistics, it was not used for training.

Finally, these results confirm that NVIDIA's Modulus implementation provides a strong foundation for research and practical adaptation. With targeted architectural and training refinements, it shows promising potential to achieve high predictive accuracy and generalization capacity if trained on the full schedule.

6. Conclusions and Future Directions

This MSc project evaluated NVIDIA's PyTorch re-implementation of GraphCast to assess its viability as a forecasting codebase. Conducted under infrastructure constraints that precluded full training schedules and larger datasets adoption, the study demonstrated that meaningful forecasting skills are achievable with this implementation. Moreover, through targeted improvements, such as data augmentation, training schedule optimization, and architectural refinements, the study demonstrated that the codebase holds substantial untapped potential to further enhance forecasting capabilities.

Review of Research Questions

1. What level of forecasting capability is achievable through this re-implementation?
    - Training a GraphCast model following the complete schedule published in [22] would have been prohibitively expensive. This project therefore prioritized computational efficiency improvements, achieving a 4× increase in the number of possible training iterations. As a result, it was possible to train models on larger data volumes and extend the forecast horizon to 12 hours. This enabled the generation of comparative results over lead times long enough to reveal meaningful performance patterns. Results are presented across multiple granularities, by region, variable, and lead time, and using domain-standard metrics. Forecasting skill varies across slices, but in most cases, the best-performing model developed in this project significantly outperformed both the original NVIDIA implementation and a domain-established baseline (i.e., FourCastNet, trained on equivalent data). Detailed forecasting capability results are provided in Section 4.11.
2. How does the predictive performance of this model compare to a reference baseline?
    - To address this question, the project adopted two reference baselines: (1) the original Modulus re-implementation of GraphCast and (2) the FourCastNet codebase. For a fair comparison, new models were trained from scratch on both codebases using the same dataset and data volumes. Performance comparisons across evaluation slices revealed considerable variability, with neither baseline consistently outperforming the other. However, through a series of targeted enhancements, the Modulus GraphCast model was substantially improved and shown to outperform both baselines across the majority of slices—spanning geographic regions, meteorological variables, and lead times. Detailed comparative results are provided in Section 4.11.

3. What effect do architectural and hyperparameter choices have on model accuracy, stability, and generalization?

    - Architectural model and training improvements, paired with carefully tuned training schedules led to statistically significant improvements in both RMSE and ACC metrics. Key modifications, such as the improvement of temporally signal smoothing, improved preparation of Sea Surface Temperature data, correction of solar irradiance misalignment, and seasonal sampling bias mitigation, contributed to enhanced model stability and generalization. An ablation study was conducted to isolate and quantify the impact of each enhancement. Detailed results are presented in Section 4.7.

Key contributions

- **Model adaptation and efficiency:** The GraphCast reimplementation was extensively refactored, resulting in a substantial speedup in both training and inference. This enabled broader experimentation and robustness checks within a constrained computational budget.
- **Forecasting skill improvements:** Domain-informed interventions, listed just above, yielded measurable improvements in predictive performance, demonstrating the value of incorporating physical insights into model design.
- **Reproducibility and benchmarking:** A robust evaluation framework was developed to ensure consistent assessment of model outputs, supporting rigorous comparison of results across model variants and facilitating future research reproducibility.
- **Codebase enhancements:** The Modulus-based GraphCast codebase was extended with inference, visualization and scoring utilities, increasing transparency and interpretability.
- **Cross-model impact:** The improved sea-surface temperature data preparation may have broader relevance beyond GraphCast; for instance, [17] attempted to utilize this channel in their training but ultimately excluded it, potentially due to the same shortcomings addressed in this study.

Methodological Summary

A robust evaluation framework was developed to enable fair and reproducible comparisons between model variants. The training and inference pipelines were designed to ensure deterministic data exposure, supporting repeatable training runs and thus results reproducibility.

Evaluation was conducted across multiple dimensions, including geographical regions, temporal lead times, and meteorological variables, allowing for granular analysis of model strengths and weaknesses. Results collection procedures were designed to minimize bias from model initialization, under-represenattion of out-of-sample data exposure, and stochastic data sampling effects. These safeguards ensure that observed performance differences are attributable to model changes rather than incidental factors.

Future Work

Several directions may extend the contributions of this work:

- **Edge and node encoding:** Explore alternative representations of the information mapping of spatial and physical variables information mapping into the GNN edge and node entities.
- **Data recency and non-stationarity:** The dataset employed in this project covers the period up to 2018. Unknown is the influence of training data recency on model generalization, particularly in the context of the evolving climate.
- **Probabilistic modeling:** Incorporate ensemble methods or uncertainty quantification techniques (e.g., CRPS-based objectives or Bayesian layers) to improve the operational utility of forecasts under uncertainty.
- **Different techniques integration:** Explore how advancements such as [15], [17], [18], [21], [23], [41]–[43] could be integrated into GraphCast to further improve its predictive abilities.

Final Remarks

This project has demonstrated that a principled, resource-aware approach can effectively uncover new information even in a highly scrutinized state-of-the-art ML model. This work exemplifies the application of MSc-level Data Science competencies acquired through this program to a high-impact, real-world domain.

References

[1] T. Hoefler, A. Calotoiu, A. Dipankar, *et al.*, *Towards Specialized Supercomputers for Climate Sciences: Computational Requirements of the Icosahedral Nonhydrostatic Weather and Climate Model*, 2024. arXiv: 2405.13043 [physics.ao-ph]. [Online]. Available: https://arxiv.org/abs/2405.13043.

[2] T. Hoefler, *ML for High-Performance Climate and Earth Virtualization Engines*, Last accessed: 2025-01-26, 2024. [Online]. Available: https://www.youtube.com/watch?v=Ky0qBEhSs3Q.

[3] J. A. Brotzge, D. Berchoff, D. L. Carlis, *et al.*, *Challenges and Opportunities in Numerical Weather Prediction*, *Bulletin of the American Meteorological Society*, vol. 104, no. 3, E698–E705, 2023. DOI: 10.1175/BAMS-D-22-0172.1. [Online]. Available: https://journals.ametsoc.org/view/journals/bams/104/3/BAMS-D-22-0172.1.xml.

[4] S. Ferebee, *Predicting Tomorrow: A Review of Machine Learning's Role in Shaping Environmental Forecasts*, Jan. 2024. DOI: 10.70389/PJS.100008. [Online]. Available: https://www.researchgate.net/publication/384780774_Predicting_Tomorrow_A_Review_of_Machine_Learning's_Role_in_Shaping_Environmental_Forecasts.

[5] P. Dueben, Last accessed: 2025-01-26, 2021. [Online]. Available: https://www.ecmwf.int/en/about/media-centre/science-blog/2021/large-scale-machine-learning-applications-weather-and.

[6] Z. Ben-Bouallegue, M. C. A. Clare, L. Magnusson, *et al.*, *The rise of data-driven weather forecasting*, 2023. arXiv: 2307.10128 [physics.ao-ph]. [Online]. Available: https://arxiv.org/abs/2307.10128.

[7] H. Hersbach, B. Bell, P. Berrisford, *et al.*, *The ERA5 global reanalysis*, *Quarterly Journal of the Royal Meteorological Society*, vol. 146, no. 730, pp. 1999–2049, 2020. DOI: https://doi.org/10.1002/qj.3803. eprint: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/qj.3803. [Online]. Available: https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/qj.3803.

[8] F. Scarselli, M. Gori, A. C. Tsoi, *et al.*, *The Graph Neural Network Model*, 2009. DOI: 10.1109/TNN.2008.2005605.

[9] R. Keisler, *Forecasting Global Weather with Graph Neural Networks*, 2022. arXiv: 2202.07575 [physics.ao-ph]. [Online]. Available: https://arxiv.org/abs/2202.07575.

[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, 2021. arXiv: 2010.11929 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2010.11929.

[11] Z. Li, N. Kovachki, K. Azizzadenesheli, *et al.*, *Fourier Neural Operator for Parametric Partial Differential Equations*, 2021. arXiv: 2010.08895 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2010.08895.

[12] S. Rasp, S. Hoyer, A. Merose, *et al.*, *WeatherBench 2: A benchmark for the next generation of data-driven global weather models*, 2024. arXiv: 2308.15560 [physics.ao-ph]. [Online]. Available: https://arxiv.org/abs/2308.15560.

[13] ECMWF, *ECMWF's AI forecasts become operational*, 2025. [Online]. Available: https://www.ecmwf.int/en/about/media-centre/news/2025/ecmwfs-ai-forecasts-become-operational.

[14] S. Lang, M. Alexe, M. Chantry, *et al.*, *AIFS – ECMWF's data-driven forecasting system*, 2024. arXiv: 2406.01465 [physics.ao-ph]. [Online]. Available: https://arxiv.org/abs/2406.01465.

[15] M. Alexe, E. Boucher, P. Lean, *et al.*, *GraphDOP: Towards skilful data-driven medium-range weather forecasts learnt and initialised directly from observations*, 2024. arXiv: 2412.15687 [physics.ao-ph]. [Online]. Available: https://arxiv.org/abs/2412.15687.

[16] T. Rackow, N. Koldunov, C. Lessig, *et al.*, *Robustness of AI-based weather forecasts in a changing climate*, 2024. arXiv: 2409.18529 [physics.ao-ph]. [Online]. Available: https://arxiv.org/abs/2409.18529.

[17] J. Pathak, S. Subramanian, P. Harrington, *et al.*, *FourCastNet: A Global Data-driven High-resolution Weather Model using Adaptive Fourier Neural Operators*, GitHub repository: https://github.com/NVlabs/FourCastNet, 2022. [Online]. Available: https://arxiv.org/abs/2202.11214.

[18] I. Price, A. Sanchez-Gonzalez, F. Alet, *et al.*, *GenCast: Diffusion-based ensemble forecasting for medium-range weather*, 2024. arXiv: 2312.15796 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2312.15796.

[19] T. Kurth, S. Subramanian, P. Harrington, *et al.*, *FourCastNet: Accelerating Global High-Resolution Weather Forecasting using Adaptive Fourier Neural Operators*, 2022. arXiv: 2208.05419 [physics.ao-ph]. [Online]. Available: https://arxiv.org/abs/2208.05419.

[20] B. Bonev, T. Kurth, C. Hundt, *et al.*, *Spherical Fourier Neural Operators: Learning Stable Dynamics on the Sphere*, 2023. arXiv: 2306.03838 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2306.03838.

[21] K. Bi, L. Xie, H. Zhang, *et al.*, *Pangu-Weather: A 3D High-Resolution Model for Fast and Accurate Global Weather Forecast*, 2022. arXiv: 2211.02556 [physics.ao-ph]. [Online]. Available: https://arxiv.org/abs/2211.02556.

[22] R. Lam, A. Sanchez-Gonzalez, M. Willson, *et al.*, *GraphCast: Learning skillful medium-range global weather forecasting*, 2023. arXiv: 2212.12794 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2212.12794.

[23] L. Chen, X. Zhong, F. Zhang, *et al.*, *FuXi: A cascade machine learning forecasting system for 15-day global weather forecast*, 2023. arXiv: 2306.12873 [physics.ao-ph]. [Online]. Available: https://arxiv.org/abs/2306.12873.

[24] X. Zhong, L. Chen, X. Fan, *et al.*, *FuXi-2.0: Advancing machine learning weather forecasting model for practical applications*, 2024. arXiv: 2409.07188 [physics.ao-ph]. [Online]. Available: https://arxiv.org/abs/2409.07188.

[25] Z. Wu, S. Pan, F. Chen, *et al.*, *A Comprehensive Survey on Graph Neural Networks*, *IEEE Transactions on Neural Networks and Learning Systems*, 2019. [Online]. Available: https://arxiv.org/abs/1901.00596.

[26] J. Gilmer, S. S. Schoenholz, P. F. Riley, *et al.*, *Neural Message Passing for Quantum Chemistry*, 2017. arXiv: 1704.01212 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1704.01212.

[27] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, *et al.*, *Learning Mesh-Based Simulation with Graph Networks*, 2021. arXiv: 2010.03409 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2010.03409.

[28] Q. Li, Z. Han, and X.-M. Wu, *Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning*, 2018. arXiv: 1801.07606 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1801.07606.

[29] L. Zhao and L. Akoglu, *PairNorm: Tackling Oversmoothing in GNNs*, 2020. arXiv: 1909.12223 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1909.12223.

[30] M. Fortunato, T. Pfaff, P. Wirnsberger, *et al.*, *MultiScale MeshGraphNets*, *arXiv preprint arXiv:2210.00612*, 2022.

[31] S. Lang, M. Alexe, M. C. A. Clare, *et al.*, *AIFS-CRPS: Ensemble forecasting using a model trained with a loss function based on the Continuous Ranked Probability Score*, 2024. arXiv: 2412.15832 [physics.ao-ph]. [Online]. Available: https://arxiv.org/abs/2412.15832.

[32] *The ERA5 subset curated for FourCastNet*, Last accessed: 2025-02-01. [Online]. Available: https://github.com/NVlabs/FourCastNet?tab=readme-ov-file#quick-links.

[33] *The ERA5 subset curated for WeatherBench2*, Last accessed: 2025-02-01. [Online]. Available: https://weatherbench2.readthedocs.io/en/latest/data-guide.html#era5.

[34] D. Lavers, A. Simmons, F. Vamborg, *et al.*, *An evaluation of ERA5 precipitation for climate monitoring*, *Quarterly Journal of the Royal Meteorological Society*, vol. 148, Aug. 2022. DOI: 10.1002/qj.4351.

[35] M. Schultz, $SPCL_Bcast52-Deeplearningcanbeatnumericalweatherprediction!What'snext?$, Last accessed: 2025-03-11, 2024. [Online]. Available: https://www.youtube.com/watch?v=i5nUrT5WNe0.

[36] ECMWF, *Forecast User Guide - Section 12.A Statistical Concepts - Deterministic Data*, Last accessed: 2025-03-11. [Online]. Available: https://confluence.ecmwf.int/x/sgJNH.

[37] ECMWF, *Forecast User Guide - Section 6.2.2 Anomaly Correlation Coefficient*, Last accessed: 2025-03-11. [Online]. Available: https://confluence.ecmwf.int/x/SP_RGw.

[38] L. Olivetti and G. Messori, *Do data-driven models beat numerical models in forecasting weather extremes? A comparison of IFS HRES, Pangu-Weather, and GraphCast*, *Geoscientific Model Development*, vol. 17, no. 21, pp. 7915–7962, 2024. DOI: 10.5194/gmd-17-7915-2024. [Online]. Available: https://gmd.copernicus.org/articles/17/7915/2024/.

[39] M. Cao, K. Mao, Y. Yan, *et al.*, *A new global gridded sea surface temperature data product based on multisource data*, *Earth System Science Data*, vol. 13, no. 5, pp. 2111–2134, 2021. DOI: 10.5194/essd-13-2111-2021. [Online]. Available: https://essd.copernicus.org/articles/13/2111/2021/.

[40] S. Siddiqui, J. Kossaifi, B. Bonev, *et al.*, *Exploring the design space of deep-learning-based weather forecasting systems*, Oct. 2024. [Online]. Available: https://arxiv.org/abs/2410.07472.

[41] T. Nguyen, J. Brandstetter, A. Kapoor, *et al.*, *ClimaX: A foundation model for weather and climate*, 2023. arXiv: 2301.10343 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2301.10343.

[42] M. Poli, S. Massaroli, E. Nguyen, *et al.*, *Hyena Hierarchy: Towards Larger Convolutional Language Models*, 2023. arXiv: 2302.10866 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2302.10866.

[43] S. Ding, J. Li, J. Wang, *et al.*, *Multi-scale Efficient Graph-Transformer for Whole Slide Image Classification*, 2023. arXiv: 2305.15773 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2305.15773.

[44] N. C. for Environmental Prediction (NCEP), *Climate Forecast System Reanalysis (CFSR)*, Accessed via https://climatedataguide.ucar.edu/climate-data/climate-forecast-system-reanalysis-cfsr, 2014.

[45] S. KOBAYASHI, Y. OTA, Y. HARADA, *et al.*, *The JRA-55 Reanalysis: General Specifications and Basic Characteristics*, *Journal of the Meteorological Society of Japan. Ser. II*, vol. 93, no. 1, pp. 5–48, 2015. DOI: 10.2151/jmsj.2015-001.

[46] R. Gelaro, W. McCarty, M. J. Suárez, *et al.*, *The Modern-Era Retrospective Analysis for Research and Applications, Version 2 (MERRA-2)*, *Journal of Climate*, vol. 30, no. 14, pp. 5419–5454, 2017. DOI: 10.1175/JCLI-D-16-0758.1.

[47] ECMWF, *Integrated Forecasting System (IFS)*, *European Centre for Medium-Range Weather Forecasts (ECMWF)*, 2024, Last accessed: 2025-02-01. [Online]. Available: https://www.ecmwf.int/en/forecasts/documentation-and-support/changes-ecmwf-model.

[48] E. C. for Medium-Range Weather Forecasts (ECMWF), *Medium-range forecasts, inclusive of High-Resolution Forecasts (HRES) and Ensemble Forecasts (ENS)*, Last accessed: 2025-02-01, 2024. [Online]. Available: https://www.ecmwf.int/en/forecasts/documentation-and-support/medium-range-forecasts.

[49] ECMWF, *AIFS Machine Learning data*. [Online]. Available: https://www.ecmwf.int/en/forecasts/dataset/aifs-machine-learning-data.

[50] N. N. W. Service, *Global Forecast System (GFS)*, *National Centers for Environmental Prediction (NCEP)*, 2024, Last accessed: 2025-02-01. [Online]. Available: https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/global-forcast-system-gfs.

[51] V. Eyring, S. Bony, G. A. Meehl, *et al.*, *Overview of the Coupled Model Intercomparison Project Phase 6 (CMIP6) experimental design and organization*, *Geoscientific Model Development*, vol. 9, pp. 1937–1958, 2016. DOI: 10.5194/gmd-9-1937-2016.

[52] N. N. C. for Environmental Information, *International Best Track Archive for Climate Stewardship*, Last accessed: 2025-02-01. [Online]. Available: https://www.ncei.noaa.gov/access/metadata/landing-page/bin/iso?id=gov.noaa.ncdc:C01552.

Appendices

A. Abbreviations and definitions

A.1 Abbreviations

Abbreviation	Definition
a.s.l.	above sea level
AIFS	The Artificial Intelligence/Integrated Forecasting System [14]
CDS	Compernicus Data Service
CNN	Convolutional Neural Network
DDP	Distributed Data Parallel
DL	Deep Learning
DNN	Deep Neural Network
ECMWF	European Centre for Medium-Range Weather Forecasts
FPR	Final Project Report
GCP	Google Cloud Platform
Git PR	Git Pull Request
GNN	Graph Neural Network
HPC	High-Performance Computing
IFS	Integrated Forecasting System, developed and operated by ECMWF
ML	Machine Learning
MSE	Mean Squared Error
NWP	Numerical Weather Prediction
RI	Research Infrastructure
RMSE	Root Mean Squared Error
wandb	Weight and Biases[1]

Table A.1: List of Abbreviations

A.2 Definitions

This section introduces common weather forecasting concepts, covering data assimilation, reanalysis, surface and pressure level variables, and probabilistic forecasting. These elements are essential in atmospheric state representations and for improving forecasts.

A.2.1 Data Assimilation

Data assimilation is the process of combining sparse real-world observations, such as data from satellites, weather stations, ships, and airplanes, with numerical models to generate a

dense representation of the state of the atmosphere. In the past, satellite data availability was limited, leading to a reduced global coverage, while today observational input data remain unevenly distributed, with the Northern Hemisphere having a higher density of observational sources compared to the Southern Hemisphere [7].

The primary goal of data assimilation is to produce a dense 3D representation of atmospheric variables, such as temperature, wind speed, and pressure, across latitude, longitude, altitude, and time. This comprehensive dataset provides the initial conditions required for numerical weather prediction (NWP) models, which use it to generate forecasts. The accuracy of these forecasts depends on the quality of the assimilated initial state [7].

A.2.2 Reanalysis

Reanalysis is the process of reconstructing past atmospheric conditions by integrating historical observational data, such as temperature, wind, and pressure measurements, into a fixed numerical weather prediction (NWP) model over extended periods [7]. Unlike operational weather forecasts that provide short-term predictions, reanalysis generates long-term, homogeneous, and physically reliable climate datasets, enabling the analysis of weather patterns over extended periods of time.

Several major reanalysis datasets have been developed and are commonly used in climate research and operational applications. Notable examples include the ERA5 from the European Centre for Medium-Range Weather Forecasts (ECMWF), of which a subset will be used as the main dataset for this project. This dataset will be described in more detail in the methods chapter.

A.2.3 Variables and Channels

Channels are the individual layers in each weather dataset sample; each channel represents a physical quantity at a specific spatial grid and vertical level, like temperature or wind at specified altitudes. Channels resemble image data input planes, encoding scalar fields on Earth's surface. Variables group these channels by common physical quantities across different levels, such as multiple temperature levels or wind components. This distinction is essential for model input/output definitions, data pipeline organization, and meaningful result interpretation.

A.2.4 Surface and Pressure Level Variables

Weather models represent atmospheric data at different vertical levels, categorized into surface and pressure levels.

Surface variables include meteorological variables measured at the Earth's surface, such as temperature, wind speed, precipitation, and pressure. These are crucial for applications like agriculture and ground transportation.

Pressure level variables represent conditions at specific atmospheric pressures (e.g., 1000 hPa, 850 hPa, 500 hPa), providing insight into vertical atmospheric structure. Pressure levels are preferred over fixed altitudes, as atmospheric pressure decreases predictably with height, offering a more consistent representation of weather patterns. Reanalysis datasets, such as ERA5, provide data across multiple pressure levels.

A.2.5 Probabilistic Forecasting

Probabilistic forecasting generates multiple potential outcomes, providing a range of scenarios instead of a single deterministic prediction. This approach accounts for uncertainties in initial conditions and the inherently chaotic nature of complex systems, making it particularly useful for extreme event prediction, such as hurricanes and heavy rainfall.

The outcome of a probabilistic forecast is a distribution of possible events, representing the range and likelihood of different scenarios. By estimating the likelihood of different outcomes, probabilistic forecasts improve risk assessment and decision-making for early warning systems and disaster management.

Common techniques include ensemble forecasting, where multiple simulations with slight variations are run to assess uncertainties, and machine learning methods, such as generative models, that can generate diverse plausible scenarios.

B. A primer on GNNs

Consider a weather prediction problem where the input data is a stack of images, each representing a different atmospheric variable or pressure level. These images form a rectangular representation of the atmosphere state, whose grid points can be mapped into nodes feature vectors. Edges would then represent the relationships between nodes, such as distance.

Model construction: The construction of a GNN model for weather prediction involves several key components:

- **Data mapping:** The gridded atmospheric data is mapped into a graph structure, where each grid point corresponds to a node with feature vectors representing variables such as temperature, humidity, and wind speed. Spatial relationships are encoded as edges, capturing dependencies between neighboring grid points (e.g., geographical barriers such as mountains).
- **Nodes and processing:** Each node in the graph corresponds to a neural network model, typically a multi-layer perceptron (MLP), which processes the node's own features and those aggregated from its neighbors. During each iteration, nodes receive input data from their neighbors through the edges, aggregate this information using predefined functions (e.g., summation or mean), and update their state accordingly.
- **Graph architecture:** A GNN model is defined with multiple layers of message passing and transformation operations. Each iteration starts by collecting neighbor information, performing aggregation, and applying shared weight matrices across all nodes. These weights are learned during training and followed by non-linear activation functions (e.g., ReLU) and normalization techniques (e.g., batch normalization) to stabilize training.
- **Iteration process:** An iteration begins with nodes collecting feature information from their neighbors, combining it with their own features, processing the combined data through the neural layers, and updating their representations. The iteration ends once the updated feature representations have been computed and the process moves to the next iteration or concludes based on a convergence criterion.
- **Processing pipeline:** The input graph undergoes preprocessing, message passing, and iterative updates to refine the node representations based on neighboring information. Fully connected layers at the output stage facilitate final predictions.

Training process: The GNN training process involves several phases:

1. **Initialization:** Each node is initialized with its feature vector derived from atmospheric data, which is loaded from files containing atmospheric data for a given point in time. This initialization step occurs before the training or inference process starts, ensuring that each node has the necessary input values for subsequent computations.

2. **Message passing:** Nodes iteratively exchange information with their neighbors, aggregating their features using predefined functions such as summation or mean. This iterative steps process neighboring information.
3. **Update:** The aggregated information is processed through the shared fully connected neural layers, applying weight transformations, activation functions (e.g., ReLU), and normalization techniques (e.g., batch normalization) to refine node states. The weights of these is commonly shared among all nodes.
4. **Optimization:** Loss is computed by comparing predicted and actual values for a given node, followed by backpropagation across the network to adjust parameters and improve performance.

Inference process: During inference, the trained GNN utilizes the learned weights to propagate information through the graph. Starting with initialized node features, the message-passing mechanism is applied iteratively to refine node representations. Once the final state is achieved, predictions for unobserved stations or future atmospheric states are extracted from the updated node features.

C. Other Weather Datasets

While the Background chapter focused on the datasets employed in this project, the present Appendix provides an overview of alternative weather-related datasets commonly cited in literature.

In the domain of reanalysis datasets, the following alternatives to ERA5 exist:

- **CFSR** (Climate Forecast System Reanalysis) [44] by NCEP, covers the period from 1979 at a temporal resolution of 6 hours and a horizontal resolution between $0.5° \times 0.5°$ and $0.25° \times 0.25°$ depending on the variable of interest and the latitude. It comprises fewer channels than ERA5. It has been used for various climate and weather prediction applications (it also includes carbon dioxide concentrations), though it is less common in ML-based forecasting due to lower spatial and temporal resolution compared to ERA5.
- **JRA-55** [45], produced by the Japan Meteorological Agency, spans 1958 to the present with a lower resolution than ERA5 ($1.25° \times 1.25°$, 288×145 grid). It is not commonly used in the ML field due to its coarser resolution and older data assimilation system.
- **MERRA-2** [46], developed by NASA's Global Modeling and Assimilation Office, covers 1980 to the present with a spatial resolution of $0.5° \times 0.625°$ (576x361 grid). Unlike ERA5, which is optimized for general weather forecasting, MERRA-2 places a stronger emphasis on aerosols and radiation processes, making it more suitable for climate and air quality research. Its coarser resolution and specific focus result in less frequent adoption for ML-based weather forecasting.

Below is a summary of other commonly cited weather-related datasets:

- **IFS** (Integrated Forecasting System) [47], developed by ECMWF, itself not a dataset but a state-of-the-art physics-driven model used to generate global weather predictions. Notably, ERA5 is derived from the IFS, using its data assimilation framework to reconstruct past atmospheric conditions. Two datasets based on IFS are commonly cited in the field of data driven weather forecasting:
  - **HRES (High-Resolution Forecast)** [48], a deterministic weather forecast produced by ECMWF, providing a single high-resolution (about 9 km) global prediction up to 10 days ahead, outputs forecasts four times daily. Past forecast data is archived in the ECMWF MARS system and access to it is limited to ECMWF Member States and researchers upon request.
  - **IFS ENS (Ensemble Prediction System)** [48], a probabilistic forecast using 51 ensemble members, running at about 18 km resolution (up to 15 days) and at about 36 km (up to 46 days), designed to quantify forecast uncertainty, outputs

forecasts twice daily. Archived ensemble forecasts are available in MARS, with limited subsets accessible through the CDS.

- **AIFS** (Artificial Intelligence Forecasting System) [14], developed by ECMWF, is a novel, experimental, and evolving forecasting system that utilizes ML techniques to generate weather predictions. It is not a dataset itself but rather a model. Two configurations are run: deterministic and probabilistic. For both configurations historical records are limited, starting from January 2023. Data access and technical details such as resolution and output frequency are described in [49].
  - **AIFS deterministic** [49] provides a single, high-resolution forecast, similar in concept to HRES but generated using data-driven methods.
  - **AIFS probabilistic** [49] provides an ensemble of forecasts, similar to the IFS ENS, but generated using data-driven methods.
- **NOAA's GFS operational forecasts** [50], produced by the U.S. National Weather Service, comprise deterministic forecasts up to 16 days ahead at 0.25° resolution. It is updated every 6 hours but is generally considered less accurate than IFS.
- **CMIP6 climate simulations** [51], part of the Coupled Model Intercomparison Project, it provides long-term climate projections rather than short-term weather forecasts. It comprises data with coarse resolutions, between 100 and 250 km. Unlike operational forecasting datasets, which comprise forecasts over days to weeks, CMIP6 focuses on multi-decade climate trends and does not aim for short-term precision. This makes it ideal for climate change research rather than day-to-day weather forecasting.
- **IBTrACS** The International Best Track Archive for Climate Stewardship (IBTrACS) [52], provides a global dataset of tropical cyclone tracks and is used for evaluating the performance of machine learning models in predicting tropical cyclone trajectories and intensity.

D. Extended Out-of-Sample Model Performance Results

This appendix contains extended results.

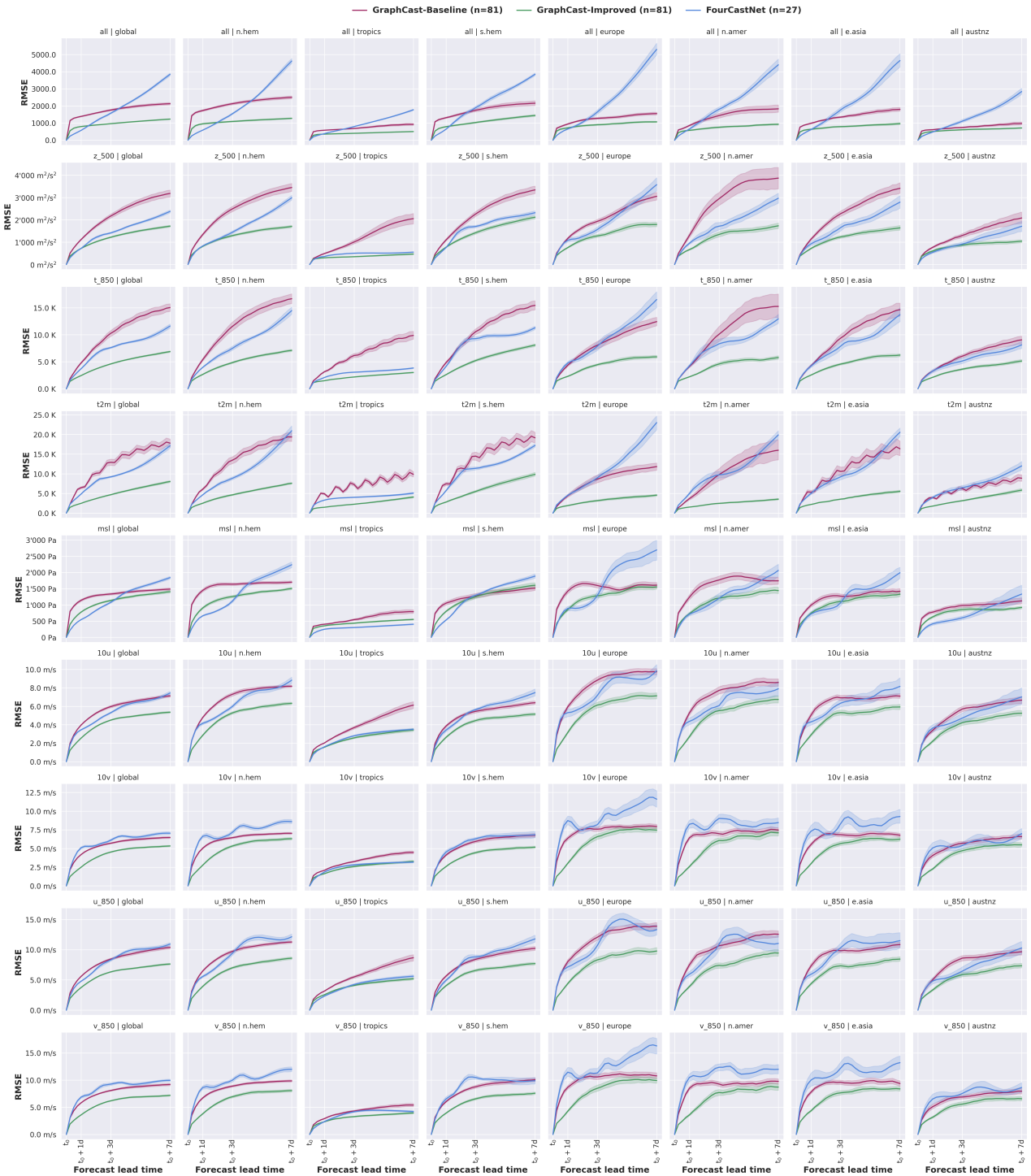


Figure D.1: Root Mean Square Error (RMSE) across variables and regions. Lower values indicate better forecast skill. Shaded regions reflect variability across inference initializations and model instances.

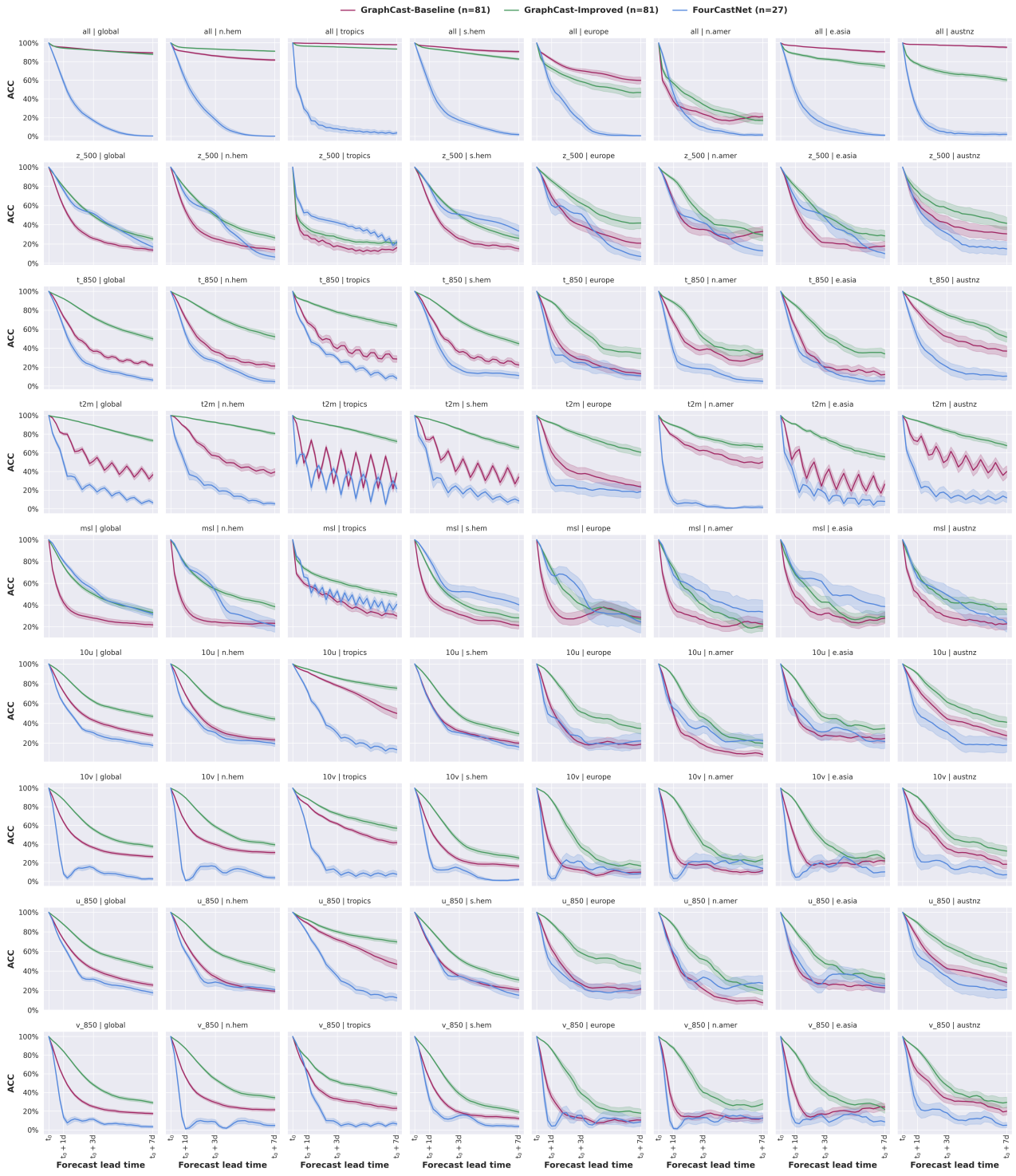


Figure D.2: Anomaly Correlation Coefficient (ACC) across variables and regions. Higher values indicate better forecast skill. Shaded regions reflect variability across inference initializations and model instances.

Variable	Model	global	n.hem	tropics	s.hem	europe	n.amer	e.asia	austnz
all	GraphCast-Baseline	1392.89 ± 121.98	1736.49 ± 223.59	584.47 ± 38.27	1317.83 ± 192.47	947.60 ± 166.16	861.78 ± 196.04	992.69 ± 146.62	616.47 ± 104.25
all	GraphCast-Improved	801.19 ± 81.45	959.85 ± 165.08	**357.46 ± 26.73**	792.30 ± 140.56	723.46 ± 160.13	**578.25 ± 131.34**	641.58 ± 141.56	506.15 ± 76.16
all	FourCastNet	**578.41 ± 21.38**	**591.66 ± 29.81**	386.33 ± 14.82	**650.17 ± 49.55**	**656.79 ± 68.39**	655.98 ± 105.42	**628.12 ± 97.22**	**482.55 ± 55.43**
z_500	GraphCast-Baseline	1121.78 ± 140.26	1346.72 ± 242.35	492.42 ± 108.32	1104.64 ± 234.28	1169.63 ± 222.00	1301.53 ± 420.41	1137.88 ± 259.71	770.89 ± 170.91
z_500	GraphCast-Improved	**723.90 ± 71.91**	**817.73 ± 140.66**	**299.95 ± 31.65**	778.96 ± 121.44	**758.07 ± 141.16**	**669.36 ± 143.54**	**720.84 ± 144.72**	606.42 ± 159.97
z_500	FourCastNet	725.49 ± 26.94	833.85 ± 59.22	361.50 ± 12.98	**761.75 ± 39.54**	1084.88 ± 210.39	950.94 ± 129.78	926.07 ± 227.22	**546.38 ± 122.78**
t_850	GraphCast-Baseline	4.82 ± 0.75	5.41 ± 1.18	3.16 ± 0.54	4.88 ± 0.92	4.36 ± 1.10	4.13 ± 0.97	4.29 ± 0.76	3.28 ± 0.60
t_850	GraphCast-Improved	**2.46 ± 0.18**	**2.67 ± 0.41**	**1.49 ± 0.08**	**2.63 ± 0.36**	**2.31 ± 0.35**	**2.22 ± 0.28**	**2.55 ± 0.37**	**2.13 ± 0.33**
t_850	FourCastNet	3.73 ± 0.14	3.98 ± 0.23	2.09 ± 0.06	4.16 ± 0.27	4.76 ± 0.83	4.23 ± 0.64	4.21 ± 0.80	3.28 ± 0.86
t2m	GraphCast-Baseline	6.48 ± 1.27	6.10 ± 1.23	4.90 ± 1.15	7.45 ± 1.88	4.39 ± 1.72	3.67 ± 1.54	5.32 ± 2.10	3.97 ± 1.11
t2m	GraphCast-Improved	**2.48 ± 0.18**	**2.51 ± 0.47**	**1.48 ± 0.10**	**2.81 ± 0.50**	**1.90 ± 0.48**	**1.52 ± 0.17**	**2.20 ± 0.25**	**1.78 ± 0.22**
t2m	FourCastNet	4.87 ± 0.22	4.66 ± 0.24	3.30 ± 0.18	5.73 ± 0.47	4.51 ± 0.54	4.97 ± 0.93	5.03 ± 0.77	3.98 ± 0.70
msl	GraphCast-Baseline	1143.56 ± 140.50	1449.48 ± 212.14	404.96 ± 49.26	1054.81 ± 260.68	1423.36 ± 323.16	1238.32 ± 335.27	993.23 ± 246.91	777.31 ± 206.89
msl	GraphCast-Improved	780.59 ± 73.51	907.46 ± 162.66	354.67 ± 35.98	803.31 ± 120.47	**856.94 ± 197.74**	727.53 ± 183.44	745.58 ± 161.24	630.30 ± 146.84
msl	FourCastNet	**547.86 ± 19.87**	**666.46 ± 47.13**	**250.77 ± 25.70**	**536.98 ± 43.63**	889.77 ± 180.92	**714.85 ± 146.52**	**721.11 ± 199.36**	**416.81 ± 113.21**
10u	GraphCast-Baseline	3.98 ± 0.21	4.81 ± 0.44	2.03 ± 0.16	3.87 ± 0.58	5.95 ± 0.98	5.29 ± 1.02	5.05 ± 1.14	3.40 ± 1.08
10u	GraphCast-Improved	**2.56 ± 0.21**	**2.87 ± 0.40**	**1.56 ± 0.09**	**2.65 ± 0.45**	**3.16 ± 0.59**	**2.71 ± 0.69**	**2.90 ± 0.66**	**2.22 ± 0.68**
10u	FourCastNet	3.46 ± 0.16	4.16 ± 0.37	1.61 ± 0.09	3.42 ± 0.46	5.31 ± 1.10	4.38 ± 0.98	4.32 ± 0.93	3.64 ± 1.05
10v	GraphCast-Baseline	4.20 ± 0.25	5.05 ± 0.52	2.09 ± 0.17	4.13 ± 0.55	6.36 ± 1.15	6.42 ± 1.10	5.54 ± 1.04	4.07 ± 1.43
10v	GraphCast-Improved	**2.51 ± 0.18**	**2.78 ± 0.35**	**1.64 ± 0.10**	**2.60 ± 0.34**	**3.03 ± 0.58**	**2.58 ± 0.61**	**2.98 ± 0.52**	**2.26 ± 0.57**
10v	FourCastNet	5.11 ± 0.38	6.64 ± 0.63	1.85 ± 0.15	4.53 ± 0.67	8.73 ± 1.55	8.25 ± 1.50	7.32 ± 2.07	5.08 ± 1.86
u_850	GraphCast-Baseline	5.60 ± 0.28	6.50 ± 0.60	3.16 ± 0.30	5.66 ± 0.78	8.02 ± 1.50	7.50 ± 1.64	7.33 ± 1.62	5.03 ± 1.26
u_850	GraphCast-Improved	**3.76 ± 0.31**	**3.95 ± 0.55**	2.57 ± 0.18	**4.07 ± 0.65**	**4.40 ± 0.84**	**3.91 ± 0.94**	**4.38 ± 1.05**	**3.35 ± 0.84**
u_850	FourCastNet	4.80 ± 0.21	5.58 ± 0.51	**2.33 ± 0.15**	4.96 ± 0.54	7.15 ± 1.51	5.96 ± 1.18	5.95 ± 1.25	4.83 ± 1.36
v_850	GraphCast-Baseline	5.84 ± 0.32	6.74 ± 0.75	2.92 ± 0.32	6.05 ± 0.69	8.67 ± 1.52	8.50 ± 1.44	7.80 ± 1.36	5.11 ± 1.88
v_850	GraphCast-Improved	**3.61 ± 0.26**	**3.82 ± 0.52**	**2.33 ± 0.18**	**3.94 ± 0.50**	**4.28 ± 0.87**	**3.60 ± 0.90**	**4.36 ± 0.83**	**3.18 ± 0.84**
v_850	FourCastNet	6.88 ± 0.55	8.65 ± 0.86	2.38 ± 0.22	6.55 ± 1.04	11.46 ± 2.17	10.50 ± 2.03	10.02 ± 3.21	6.53 ± 2.44

Table D.1: RMSE metric scores (mean ± standard deviation) for 1 day forecast lead time. For `GraphCast-Baseline` and `GraphCast-Improved` $n = 81$, and for `FourCastNet` $n = 27$. Lower values are better.

Variable	Model	global	n.hem	tropics	s.hem	europe	n.amer	e.asia	austnz
all	GraphCast-Baseline	**0.95 ± 0.01**	0.91 ± 0.02	**0.99**	**0.96 ± 0.01**	**0.82 ± 0.06**	0.37 ± 0.21	**0.97 ± 0.01**	**0.98 ± 0.01**
all	GraphCast-Improved	0.94 ± 0.01	**0.95 ± 0.02**	0.96 ± 0.01	0.94 ± 0.02	0.73 ± 0.14	**0.57 ± 0.14**	0.88 ± 0.05	0.78 ± 0.07
all	FourCastNet	0.58 ± 0.07	0.60 ± 0.12	0.25 ± 0.10	0.55 ± 0.16	0.53 ± 0.19	0.46 ± 0.19	0.48 ± 0.17	0.28 ± 0.11
z_500	GraphCast-Baseline	0.59 ± 0.08	0.54 ± 0.16	0.29 ± 0.21	0.63 ± 0.10	0.69 ± 0.20	0.64 ± 0.19	0.57 ± 0.20	0.63 ± 0.29
z_500	GraphCast-Improved	**0.79 ± 0.04**	**0.79 ± 0.06**	0.36 ± 0.15	**0.80 ± 0.05**	**0.86 ± 0.11**	**0.87 ± 0.08**	**0.80 ± 0.12**	**0.75 ± 0.24**
z_500	FourCastNet	0.76 ± 0.05	0.71 ± 0.10	**0.53 ± 0.08**	0.79 ± 0.07	0.60 ± 0.22	0.63 ± 0.15	0.67 ± 0.14	0.59 ± 0.20
t_850	GraphCast-Baseline	0.73 ± 0.07	0.71 ± 0.11	0.67 ± 0.11	0.75 ± 0.07	0.59 ± 0.25	0.67 ± 0.15	0.62 ± 0.17	0.79 ± 0.13
t_850	GraphCast-Improved	**0.92 ± 0.01**	**0.92 ± 0.02**	**0.89 ± 0.04**	**0.92 ± 0.02**	**0.89 ± 0.05**	**0.88 ± 0.07**	**0.85 ± 0.08**	**0.91 ± 0.04**
t_850	FourCastNet	0.62 ± 0.07	0.55 ± 0.10	0.55 ± 0.07	0.66 ± 0.11	0.39 ± 0.21	0.36 ± 0.18	0.47 ± 0.15	0.51 ± 0.18
t2m	GraphCast-Baseline	0.80 ± 0.07	0.87 ± 0.06	0.59 ± 0.14	0.74 ± 0.13	0.61 ± 0.28	0.78 ± 0.09	0.60 ± 0.26	0.72 ± 0.19
t2m	GraphCast-Improved	**0.97 ± 0.01**	**0.98 ± 0.01**	**0.95 ± 0.02**	**0.96 ± 0.01**	**0.92 ± 0.04**	**0.89 ± 0.04**	**0.91 ± 0.04**	**0.94 ± 0.03**
t2m	FourCastNet	0.49 ± 0.09	0.47 ± 0.14	0.43 ± 0.08	0.47 ± 0.18	0.28 ± 0.19	0.08 ± 0.12	0.31 ± 0.25	0.32 ± 0.17
msl	GraphCast-Baseline	0.42 ± 0.13	0.37 ± 0.18	0.58 ± 0.09	0.47 ± 0.14	0.39 ± 0.28	0.38 ± 0.25	0.48 ± 0.25	0.51 ± 0.19
msl	GraphCast-Improved	0.76 ± 0.04	**0.77 ± 0.05**	**0.72 ± 0.07**	0.72 ± 0.06	**0.73 ± 0.14**	**0.74 ± 0.15**	0.68 ± 0.17	**0.70 ± 0.15**
msl	FourCastNet	**0.80 ± 0.04**	0.76 ± 0.09	0.65 ± 0.08	**0.82 ± 0.07**	0.68 ± 0.18	0.68 ± 0.18	**0.69 ± 0.18**	0.65 ± 0.18
10u	GraphCast-Baseline	0.72 ± 0.04	0.69 ± 0.05	0.92 ± 0.02	0.63 ± 0.05	0.55 ± 0.19	0.49 ± 0.17	0.51 ± 0.17	0.76 ± 0.15
10u	GraphCast-Improved	**0.89 ± 0.02**	**0.89 ± 0.02**	**0.95 ± 0.01**	**0.83 ± 0.04**	**0.88 ± 0.06**	**0.87 ± 0.08**	**0.84 ± 0.06**	**0.90 ± 0.06**
10u	FourCastNet	0.60 ± 0.04	0.56 ± 0.06	0.72 ± 0.05	0.62 ± 0.05	0.47 ± 0.22	0.55 ± 0.18	0.55 ± 0.22	0.51 ± 0.25
10v	GraphCast-Baseline	0.64 ± 0.05	0.63 ± 0.05	0.82 ± 0.03	0.58 ± 0.07	0.35 ± 0.19	0.33 ± 0.20	0.45 ± 0.19	0.67 ± 0.22
10v	GraphCast-Improved	**0.87 ± 0.02**	**0.90 ± 0.02**	**0.89 ± 0.02**	**0.83 ± 0.03**	**0.87 ± 0.05**	**0.89 ± 0.05**	**0.86 ± 0.05**	**0.90 ± 0.05**
10v	FourCastNet	0.08 ± 0.06	0.01 ± 0.02	0.54 ± 0.05	0.26 ± 0.11	0.03 ± 0.07	0.03 ± 0.10	0.04 ± 0.09	0.26 ± 0.27
u_850	GraphCast-Baseline	0.72 ± 0.04	0.68 ± 0.05	0.89 ± 0.04	0.67 ± 0.04	0.62 ± 0.19	0.55 ± 0.18	0.47 ± 0.21	0.74 ± 0.16
u_850	GraphCast-Improved	**0.87 ± 0.03**	**0.89 ± 0.02**	**0.92 ± 0.03**	**0.83 ± 0.04**	**0.89 ± 0.06**	**0.87 ± 0.08**	**0.80 ± 0.10**	**0.89 ± 0.07**
u_850	FourCastNet	0.65 ± 0.04	0.59 ± 0.06	0.78 ± 0.04	0.69 ± 0.04	0.47 ± 0.21	0.59 ± 0.15	0.56 ± 0.22	0.57 ± 0.24
v_850	GraphCast-Baseline	0.56 ± 0.05	0.57 ± 0.05	0.63 ± 0.07	0.54 ± 0.06	0.35 ± 0.20	0.26 ± 0.22	0.37 ± 0.20	0.62 ± 0.23
v_850	GraphCast-Improved	**0.84 ± 0.03**	**0.87 ± 0.02**	**0.76 ± 0.03**	**0.81 ± 0.03**	**0.86 ± 0.05**	**0.88 ± 0.07**	**0.82 ± 0.08**	**0.86 ± 0.06**
v_850	FourCastNet	0.13 ± 0.07	0.01 ± 0.03	0.56 ± 0.06	0.32 ± 0.13	0.04 ± 0.10	0.01 ± 0.03	0.07 ± 0.13	0.26 ± 0.29

Table D.2: ACC metric scores (mean ± standard deviation) for 1 day forecast lead time. For `GraphCast-Baseline` and `GraphCast-Improved` $n = 81$, and for `FourCastNet` $n = 27$. Higher values are better.

Variable	Model	global	n.hem	tropics	s.hem	europe	n.amer	e.asia	austnz
all	GraphCast-Baseline	1763.36 ± 217.31	2133.14 ± 295.54	688.35 ± 96.44	1751.01 ± 382.69	1278.67 ± 235.86	1440.10 ± 613.63	1350.17 ± 220.98	746.25 ± 156.75
all	GraphCast-Improved	**968.35 ± 86.19**	**1097.43 ± 161.01**	**406.10 ± 42.66**	**1036.54 ± 187.04**	**906.27 ± 175.71**	**757.33 ± 197.57**	**795.09 ± 172.88**	**613.26 ± 116.30**
all	FourCastNet	1581.82 ± 110.69	1549.17 ± 119.10	798.19 ± 33.61	1903.62 ± 254.61	1648.57 ± 327.76	1806.45 ± 339.56	1775.10 ± 374.83	1114.68 ± 221.29
z_500	GraphCast-Baseline	2171.75 ± 401.09	2492.44 ± 554.35	1009.14 ± 381.87	2265.88 ± 497.11	1933.33 ± 400.83	2945.38 ± 1436.77	2277.33 ± 545.27	1301.98 ± 398.96
z_500	GraphCast-Improved	**1222.12 ± 130.34**	**1294.07 ± 195.10**	**350.52 ± 53.71**	**1425.09 ± 281.68**	**1280.59 ± 297.41**	**1346.07 ± 509.17**	**1235.26 ± 271.32**	**872.84 ± 277.21**
z_500	FourCastNet	1417.00 ± 84.03	1444.09 ± 160.14	505.44 ± 25.56	1704.11 ± 145.02	1567.73 ± 290.42	1716.17 ± 432.95	1654.37 ± 394.79	918.29 ± 229.74
t_850	GraphCast-Baseline	10.27 ± 2.46	11.55 ± 3.45	6.16 ± 1.67	10.58 ± 2.56	8.07 ± 3.08	10.10 ± 6.16	9.39 ± 3.14	6.04 ± 1.39
t_850	GraphCast-Improved	**4.60 ± 0.40**	**4.85 ± 0.70**	**2.18 ± 0.17**	**5.23 ± 0.93**	**4.46 ± 0.90**	**4.59 ± 1.36**	**4.57 ± 0.83**	**3.63 ± 0.72**
t_850	FourCastNet	7.62 ± 0.41	7.49 ± 1.01	3.07 ± 0.12	9.28 ± 0.97	7.89 ± 2.00	8.32 ± 2.42	7.90 ± 1.89	5.14 ± 1.31
t2m	GraphCast-Baseline	12.91 ± 3.62	13.46 ± 4.13	7.27 ± 1.91	14.47 ± 4.77	8.65 ± 4.44	9.77 ± 7.35	10.59 ± 5.11	6.25 ± 2.14
t2m	GraphCast-Improved	**4.67 ± 0.55**	**4.63 ± 0.83**	**2.25 ± 0.28**	**5.50 ± 1.28**	**3.17 ± 0.96**	**2.47 ± 0.53**	**3.54 ± 0.56**	**3.15 ± 0.57**
t2m	FourCastNet	9.22 ± 0.48	8.85 ± 0.98	4.01 ± 0.23	11.39 ± 0.98	8.97 ± 1.75	9.74 ± 2.29	9.23 ± 1.95	6.45 ± 1.43
msl	GraphCast-Baseline	1336.49 ± 143.56	1637.23 ± 183.71	546.59 ± 116.55	1283.75 ± 330.03	1598.42 ± 356.49	1769.14 ± 536.92	1269.38 ± 304.19	981.51 ± 274.57
msl	GraphCast-Improved	1136.74 ± 108.35	1255.85 ± 168.50	437.21 ± 60.01	**1252.25 ± 268.84**	**1281.14 ± 282.48**	1208.84 ± 343.18	1104.69 ± 297.41	856.54 ± 239.79
msl	FourCastNet	**1079.62 ± 68.92**	**1116.42 ± 124.50**	**303.46 ± 23.91**	1297.39 ± 109.19	1292.12 ± 336.68	**1206.08 ± 386.16**	**1071.00 ± 285.86**	**601.90 ± 151.32**
10u	GraphCast-Baseline	5.97 ± 0.59	7.31 ± 0.87	3.69 ± 0.67	5.46 ± 0.76	8.85 ± 1.63	7.73 ± 1.79	6.92 ± 1.25	5.71 ± 1.37
10u	GraphCast-Improved	**4.51 ± 0.25**	**5.29 ± 0.58**	**2.58 ± 0.36**	**4.46 ± 0.65**	**6.14 ± 1.46**	**5.53 ± 1.44**	**5.32 ± 1.31**	**4.23 ± 1.44**
10u	FourCastNet	5.31 ± 0.24	5.99 ± 0.42	2.81 ± 0.20	5.66 ± 0.31	7.19 ± 1.40	6.08 ± 1.46	6.35 ± 1.08	4.73 ± 1.37
10v	GraphCast-Baseline	5.71 ± 0.40	6.45 ± 0.57	3.25 ± 0.42	5.95 ± 0.85	7.58 ± 1.40	6.98 ± 1.50	6.85 ± 1.20	5.70 ± 1.20
10v	GraphCast-Improved	**4.53 ± 0.30**	**5.33 ± 0.65**	**2.59 ± 0.42**	**4.45 ± 0.62**	**6.59 ± 1.72**	**6.07 ± 1.73**	**5.67 ± 1.30**	**4.71 ± 1.28**
10v	FourCastNet	6.11 ± 0.37	7.17 ± 0.70	2.82 ± 0.25	6.31 ± 0.45	8.19 ± 1.40	9.02 ± 1.51	8.34 ± 2.45	5.40 ± 1.22
u_850	GraphCast-Baseline	8.45 ± 0.75	9.72 ± 1.04	5.50 ± 0.98	8.42 ± 1.07	12.07 ± 2.29	10.66 ± 2.42	9.83 ± 1.55	8.61 ± 2.00
u_850	GraphCast-Improved	**6.33 ± 0.34**	**7.01 ± 0.77**	**4.08 ± 0.46**	**6.58 ± 0.94**	**8.32 ± 2.04**	**7.40 ± 2.02**	**7.35 ± 1.71**	**5.90 ± 1.66**
u_850	FourCastNet	8.31 ± 0.55	9.42 ± 0.94	4.24 ± 0.39	8.82 ± 0.81	10.71 ± 2.11	10.89 ± 2.69	10.34 ± 1.98	6.46 ± 1.91
v_850	GraphCast-Baseline	8.09 ± 0.55	8.89 ± 0.80	4.30 ± 0.77	8.78 ± 1.23	10.59 ± 2.36	9.28 ± 2.06	9.61 ± 1.83	6.89 ± 1.63
v_850	GraphCast-Improved	**6.19 ± 0.37**	**6.94 ± 0.90**	**3.28 ± 0.41**	**6.56 ± 0.89**	**8.50 ± 2.49**	**7.72 ± 2.42**	**7.66 ± 1.81**	**5.58 ± 1.84**
v_850	FourCastNet	9.26 ± 0.63	9.99 ± 0.88	4.18 ± 0.35	10.49 ± 0.93	11.32 ± 2.21	12.37 ± 1.72	12.22 ± 3.04	7.55 ± 1.52

Table D.3: RMSE metric scores (mean ± standard deviation) for 3 days forecast lead time. For `GraphCast-Baseline` and `GraphCast-Improved` $n = 81$, and for `FourCastNet` $n = 27$. Lower values are better.

Variable	Model	global	n.hem	tropics	s.hem	europe	n.amer	e.asia	austnz
all	GraphCast-Baseline	**0.92 ± 0.02**	0.86 ± 0.03	**0.99 ± 0.01**	**0.94 ± 0.03**	**0.70 ± 0.11**	0.24 ± 0.20	**0.94 ± 0.02**	**0.97 ± 0.01**
all	GraphCast-Improved	0.92 ± 0.01	**0.93 ± 0.02**	0.96 ± 0.01	0.90 ± 0.03	0.59 ± 0.18	**0.34 ± 0.23**	0.83 ± 0.06	0.68 ± 0.10
all	FourCastNet	0.17 ± 0.05	0.18 ± 0.08	0.08 ± 0.09	0.17 ± 0.09	0.16 ± 0.12	0.10 ± 0.11	0.13 ± 0.15	0.04 ± 0.08
z_500	GraphCast-Baseline	0.26 ± 0.09	0.25 ± 0.14	0.18 ± 0.18	0.25 ± 0.15	0.39 ± 0.24	0.35 ± 0.26	0.21 ± 0.22	0.42 ± 0.32
z_500	GraphCast-Improved	0.49 ± 0.07	0.50 ± 0.11	0.27 ± 0.14	0.50 ± 0.09	**0.62 ± 0.25**	**0.49 ± 0.28**	**0.49 ± 0.24**	**0.57 ± 0.29**
z_500	FourCastNet	**0.52 ± 0.07**	**0.52 ± 0.10**	**0.42 ± 0.08**	**0.51 ± 0.13**	0.51 ± 0.24	0.42 ± 0.19	0.47 ± 0.21	0.31 ± 0.21
t_850	GraphCast-Baseline	0.37 ± 0.11	0.36 ± 0.15	0.39 ± 0.17	0.36 ± 0.13	0.28 ± 0.25	0.38 ± 0.23	0.20 ± 0.22	0.52 ± 0.26
t_850	GraphCast-Improved	**0.72 ± 0.06**	**0.74 ± 0.07**	**0.78 ± 0.08**	**0.69 ± 0.08**	**0.59 ± 0.19**	**0.51 ± 0.24**	**0.53 ± 0.19**	**0.75 ± 0.13**
t_850	FourCastNet	0.21 ± 0.06	0.26 ± 0.07	0.29 ± 0.06	0.18 ± 0.11	0.25 ± 0.19	0.18 ± 0.14	0.18 ± 0.14	0.20 ± 0.18
t2m	GraphCast-Baseline	0.51 ± 0.15	0.57 ± 0.16	0.43 ± 0.16	0.46 ± 0.20	0.37 ± 0.31	0.63 ± 0.22	0.36 ± 0.27	0.52 ± 0.27
t2m	GraphCast-Improved	**0.89 ± 0.03**	**0.93 ± 0.03**	**0.88 ± 0.04**	**0.85 ± 0.05**	**0.79 ± 0.12**	**0.76 ± 0.10**	**0.76 ± 0.10**	**0.84 ± 0.10**
t2m	FourCastNet	0.23 ± 0.07	0.23 ± 0.10	0.34 ± 0.08	0.21 ± 0.12	0.22 ± 0.19	0.04 ± 0.05	0.16 ± 0.17	0.15 ± 0.14
msl	GraphCast-Baseline	0.28 ± 0.11	0.25 ± 0.14	0.43 ± 0.12	0.31 ± 0.15	0.29 ± 0.28	0.26 ± 0.21	0.32 ± 0.23	0.29 ± 0.25
msl	GraphCast-Improved	0.50 ± 0.09	**0.55 ± 0.09**	**0.61 ± 0.09**	0.43 ± 0.14	0.41 ± 0.27	0.40 ± 0.26	0.43 ± 0.25	0.45 ± 0.23
msl	FourCastNet	**0.54 ± 0.08**	0.53 ± 0.14	0.53 ± 0.07	**0.52 ± 0.15**	**0.53 ± 0.26**	**0.52 ± 0.22**	**0.57 ± 0.23**	**0.49 ± 0.23**
10u	GraphCast-Baseline	0.43 ± 0.08	0.35 ± 0.11	0.78 ± 0.06	0.32 ± 0.09	0.20 ± 0.17	0.18 ± 0.20	0.28 ± 0.19	0.44 ± 0.20
10u	GraphCast-Improved	**0.62 ± 0.05**	**0.61 ± 0.06**	**0.86 ± 0.05**	**0.49 ± 0.08**	**0.51 ± 0.21**	**0.46 ± 0.22**	**0.46 ± 0.20**	**0.61 ± 0.22**
10u	FourCastNet	0.31 ± 0.07	0.31 ± 0.10	0.30 ± 0.09	0.30 ± 0.07	0.26 ± 0.20	0.37 ± 0.20	0.31 ± 0.24	0.31 ± 0.21
10v	GraphCast-Baseline	0.36 ± 0.07	0.40 ± 0.08	0.62 ± 0.06	0.23 ± 0.10	0.12 ± 0.14	0.18 ± 0.18	0.18 ± 0.18	0.35 ± 0.23
10v	GraphCast-Improved	**0.56 ± 0.06**	**0.59 ± 0.07**	**0.72 ± 0.08**	**0.43 ± 0.11**	**0.37 ± 0.23**	**0.39 ± 0.25**	**0.41 ± 0.21**	**0.51 ± 0.26**
10v	FourCastNet	0.15 ± 0.06	0.16 ± 0.08	0.11 ± 0.09	0.14 ± 0.10	0.21 ± 0.18	0.22 ± 0.20	0.18 ± 0.16	0.18 ± 0.16
u_850	GraphCast-Baseline	0.42 ± 0.08	0.34 ± 0.11	0.72 ± 0.09	0.35 ± 0.09	0.27 ± 0.20	0.21 ± 0.21	0.27 ± 0.19	0.42 ± 0.21
u_850	GraphCast-Improved	**0.62 ± 0.06**	**0.60 ± 0.07**	**0.80 ± 0.07**	**0.52 ± 0.08**	**0.58 ± 0.21**	**0.52 ± 0.22**	**0.47 ± 0.21**	**0.65 ± 0.20**
u_850	FourCastNet	0.32 ± 0.06	0.29 ± 0.10	0.35 ± 0.08	0.35 ± 0.09	0.25 ± 0.18	0.34 ± 0.20	0.31 ± 0.24	0.34 ± 0.23
v_850	GraphCast-Baseline	0.25 ± 0.07	0.29 ± 0.08	0.33 ± 0.11	0.19 ± 0.09	0.12 ± 0.14	0.16 ± 0.19	0.17 ± 0.17	0.33 ± 0.23
v_850	GraphCast-Improved	**0.48 ± 0.06**	**0.54 ± 0.07**	**0.52 ± 0.08**	**0.39 ± 0.11**	**0.39 ± 0.26**	**0.40 ± 0.26**	**0.41 ± 0.22**	**0.47 ± 0.25**
v_850	FourCastNet	0.11 ± 0.05	0.08 ± 0.07	0.07 ± 0.07	0.15 ± 0.09	0.16 ± 0.15	0.13 ± 0.17	0.12 ± 0.15	0.18 ± 0.16

Table D.4: ACC metric scores (mean ± standard deviation) for 3 days forecast lead time. For `GraphCast-Baseline` and `GraphCast-Improved` $n = 81$, and for `FourCastNet` $n = 27$. Higher values are better.

Variable	Model	global	n.hem	tropics	s.hem	europe	n.amer	e.asia	austnz
all	GraphCast-Baseline	2133.43 ± 322.02	2503.11 ± 363.07	927.46 ± 265.62	2160.69 ± 625.49	1556.20 ± 355.93	1832.79 ± 980.07	1802.47 ± 501.08	979.21 ± 393.86
all	GraphCast-Improved	**1234.47 ± 121.65**	**1279.90 ± 159.22**	**506.86 ± 71.98**	**1444.74 ± 249.84**	**1073.33 ± 190.08**	**932.00 ± 230.43**	**970.37 ± 267.26**	**717.53 ± 118.62**
all	FourCastNet	3846.41 ± 212.51	4621.02 ± 393.23	1777.57 ± 102.43	3847.10 ± 207.76	5298.84 ± 900.89	4402.00 ± 878.66	4649.71 ± 1042.50	2841.05 ± 489.26
z_500	GraphCast-Baseline	3181.93 ± 706.21	3447.51 ± 786.50	2050.72 ± 1063.91	3340.44 ± 809.85	3054.22 ± 802.87	3867.26 ± 2236.54	3410.47 ± 1081.18	2085.23 ± 1099.21
z_500	GraphCast-Improved	**1718.22 ± 200.75**	**1706.17 ± 239.15**	**469.51 ± 126.19**	**2117.33 ± 341.81**	**1792.40 ± 464.07**	**1733.09 ± 552.87**	**1641.04 ± 456.13**	**1043.51 ± 293.28**
z_500	FourCastNet	2377.16 ± 178.13	2988.43 ± 280.57	555.54 ± 39.71	2319.01 ± 223.07	3574.71 ± 784.75	2966.21 ± 606.76	2791.10 ± 744.16	1712.08 ± 590.36
t_850	GraphCast-Baseline	15.03 ± 3.44	16.66 ± 4.21	9.88 ± 3.52	15.43 ± 3.90	12.43 ± 3.81	15.24 ± 10.30	14.63 ± 5.69	9.05 ± 3.40
t_850	GraphCast-Improved	**6.88 ± 0.72**	**7.11 ± 0.91**	**3.02 ± 0.39**	**8.09 ± 1.14**	**5.91 ± 1.23**	**5.79 ± 1.48**	**6.23 ± 1.22**	**5.15 ± 1.19**
t_850	FourCastNet	11.61 ± 1.05	14.47 ± 1.87	3.84 ± 0.28	11.29 ± 0.87	16.51 ± 3.89	12.93 ± 2.19	13.72 ± 3.42	8.12 ± 1.74
t2m	GraphCast-Baseline	17.80 ± 4.65	19.35 ± 5.03	9.85 ± 3.22	19.15 ± 6.65	11.81 ± 4.48	15.98 ± 12.20	16.36 ± 8.47	8.88 ± 3.91
t2m	GraphCast-Improved	**8.03 ± 1.23**	**7.56 ± 1.02**	**4.07 ± 1.01**	**9.85 ± 2.09**	**4.56 ± 1.14**	**3.53 ± 0.90**	**5.54 ± 1.37**	**5.83 ± 1.74**
t2m	FourCastNet	17.11 ± 1.68	20.92 ± 3.31	5.08 ± 0.56	17.19 ± 1.18	22.99 ± 4.87	19.88 ± 2.80	20.54 ± 3.09	11.99 ± 2.91
msl	GraphCast-Baseline	1486.42 ± 164.36	1701.33 ± 192.62	791.68 ± 249.53	**1516.69 ± 363.14**	1608.44 ± 386.56	1742.77 ± 507.54	1421.33 ± 365.76	1124.79 ± 434.31
msl	GraphCast-Improved	**1408.20 ± 178.85**	**1505.78 ± 166.12**	552.12 ± 89.09	1605.14 ± 368.45	**1555.21 ± 344.63**	**1441.23 ± 453.83**	**1327.70 ± 362.25**	**921.14 ± 204.54**
msl	FourCastNet	1840.68 ± 107.67	2237.80 ± 230.43	**404.04 ± 44.08**	1886.55 ± 172.54	2693.36 ± 771.28	2062.39 ± 550.69	1991.21 ± 486.77	1333.63 ± 718.98
10u	GraphCast-Baseline	7.14 ± 0.68	8.18 ± 0.62	6.13 ± 1.70	6.39 ± 0.89	9.75 ± 1.56	8.59 ± 1.83	7.11 ± 1.15	6.68 ± 1.72
10u	GraphCast-Improved	**5.36 ± 0.45**	**6.33 ± 0.62**	**3.44 ± 0.74**	**5.15 ± 0.68**	**7.14 ± 1.57**	**6.75 ± 1.78**	**5.93 ± 1.31**	**5.25 ± 1.35**
10u	FourCastNet	7.44 ± 0.55	8.84 ± 0.97	3.50 ± 0.29	7.49 ± 1.04	9.87 ± 1.75	7.91 ± 2.12	8.17 ± 2.22	7.02 ± 2.42
10v	GraphCast-Baseline	6.47 ± 0.60	7.03 ± 0.55	4.48 ± 0.82	6.77 ± 1.13	7.95 ± 1.44	7.47 ± 1.44	6.77 ± 1.21	6.59 ± 1.46
10v	GraphCast-Improved	**5.34 ± 0.46**	**6.31 ± 0.74**	3.27 ± 0.64	**5.18 ± 0.71**	**7.46 ± 1.51**	**7.09 ± 1.95**	**6.25 ± 1.24**	**5.49 ± 1.40**
10v	FourCastNet	7.04 ± 0.51	8.60 ± 0.74	**3.18 ± 0.24**	6.84 ± 1.13	11.60 ± 2.84	8.46 ± 1.70	9.28 ± 2.50	6.99 ± 1.82
u_850	GraphCast-Baseline	10.39 ± 1.16	11.28 ± 0.95	8.68 ± 2.43	10.22 ± 1.49	13.89 ± 2.62	12.57 ± 2.43	10.85 ± 2.01	9.66 ± 2.37
u_850	GraphCast-Improved	**7.63 ± 0.63**	**8.59 ± 0.95**	**5.20 ± 0.86**	**7.70 ± 0.99**	**9.79 ± 2.63**	**9.45 ± 2.48**	**8.45 ± 1.77**	**7.35 ± 1.72**
u_850	FourCastNet	10.93 ± 0.71	12.14 ± 1.20	5.62 ± 0.57	11.78 ± 1.58	13.31 ± 2.42	11.05 ± 3.39	11.50 ± 3.60	10.29 ± 2.88
v_850	GraphCast-Baseline	9.20 ± 1.00	9.86 ± 0.95	5.41 ± 1.33	10.09 ± 1.62	10.79 ± 2.51	9.77 ± 2.24	9.38 ± 2.33	7.96 ± 2.43
v_850	GraphCast-Improved	**7.20 ± 0.65**	**8.07 ± 1.10**	**3.95 ± 0.70**	**7.59 ± 1.08**	**9.93 ± 2.29**	**8.72 ± 2.85**	**8.39 ± 1.94**	**6.57 ± 1.82**
v_850	FourCastNet	9.96 ± 0.61	11.99 ± 1.23	4.22 ± 0.41	9.90 ± 1.59	16.31 ± 3.91	11.99 ± 2.22	13.22 ± 3.31	8.64 ± 2.36

Table D.5: RMSE metric scores (mean ± standard deviation) for 7 days forecast lead time. For `GraphCast-Baseline` and `GraphCast-Improved` $n = 81$, and for `FourCastNet` $n = 27$. Lower values are better.

Variable	Model	global	n.hem	tropics	s.hem	europe	n.amer	e.asia	austnz
all	GraphCast-Baseline	**0.89 ± 0.03**	0.82 ± 0.04	**0.98 ± 0.01**	**0.91 ± 0.05**	**0.60 ± 0.17**	**0.21 ± 0.19**	**0.90 ± 0.05**	**0.95 ± 0.04**
all	GraphCast-Improved	0.88 ± 0.02	**0.91 ± 0.02**	0.93 ± 0.02	0.83 ± 0.05	0.47 ± 0.23	0.17 ± 0.19	0.75 ± 0.11	0.60 ± 0.09
all	FourCastNet	0.00 ± 0.01	0.00	0.04 ± 0.05	0.02 ± 0.04	0.01 ± 0.02	0.01 ± 0.04	0.01 ± 0.03	0.02 ± 0.05
z_500	GraphCast-Baseline	0.14 ± 0.09	0.14 ± 0.13	0.16 ± 0.17	0.15 ± 0.13	0.21 ± 0.24	**0.33 ± 0.25**	0.18 ± 0.22	0.31 ± 0.32
z_500	GraphCast-Improved	**0.25 ± 0.11**	**0.26 ± 0.12**	0.21 ± 0.11	0.26 ± 0.15	**0.42 ± 0.27**	0.29 ± 0.28	**0.28 ± 0.26**	**0.41 ± 0.32**
z_500	FourCastNet	0.17 ± 0.12	0.06 ± 0.09	**0.22 ± 0.07**	**0.33 ± 0.16**	0.07 ± 0.12	0.13 ± 0.15	0.10 ± 0.15	0.15 ± 0.18
t_850	GraphCast-Baseline	0.22 ± 0.09	0.21 ± 0.15	0.29 ± 0.16	0.22 ± 0.14	0.13 ± 0.16	0.33 ± 0.23	0.12 ± 0.17	0.37 ± 0.27
t_850	GraphCast-Improved	**0.50 ± 0.10**	**0.52 ± 0.14**	**0.63 ± 0.11**	**0.45 ± 0.11**	**0.34 ± 0.24**	**0.34 ± 0.25**	**0.34 ± 0.21**	**0.52 ± 0.24**
t_850	FourCastNet	0.06 ± 0.05	0.05 ± 0.06	0.08 ± 0.06	0.11 ± 0.09	0.11 ± 0.14	0.05 ± 0.07	0.05 ± 0.11	0.10 ± 0.11
t2m	GraphCast-Baseline	0.36 ± 0.16	0.39 ± 0.18	0.38 ± 0.14	0.34 ± 0.21	0.23 ± 0.24	0.50 ± 0.27	0.26 ± 0.22	0.40 ± 0.28
t2m	GraphCast-Improved	**0.73 ± 0.07**	**0.81 ± 0.07**	**0.72 ± 0.10**	**0.66 ± 0.09**	**0.61 ± 0.20**	**0.66 ± 0.15**	**0.56 ± 0.16**	**0.68 ± 0.15**
t2m	FourCastNet	0.07 ± 0.06	0.05 ± 0.05	0.21 ± 0.11	0.08 ± 0.08	0.18 ± 0.18	0.02 ± 0.05	0.08 ± 0.13	0.12 ± 0.16
msl	GraphCast-Baseline	0.22 ± 0.12	0.23 ± 0.13	0.30 ± 0.13	0.21 ± 0.16	**0.29 ± 0.28**	0.23 ± 0.21	0.28 ± 0.24	0.23 ± 0.25
msl	GraphCast-Improved	**0.33 ± 0.13**	**0.39 ± 0.13**	**0.49 ± 0.10**	0.28 ± 0.19	0.27 ± 0.27	0.21 ± 0.22	0.29 ± 0.24	**0.36 ± 0.24**
msl	FourCastNet	0.32 ± 0.10	0.21 ± 0.16	0.41 ± 0.08	**0.40 ± 0.16**	0.24 ± 0.27	**0.34 ± 0.29**	**0.39 ± 0.23**	0.24 ± 0.23
10u	GraphCast-Baseline	0.28 ± 0.08	0.23 ± 0.08	0.50 ± 0.25	0.20 ± 0.10	0.19 ± 0.20	0.09 ± 0.13	0.25 ± 0.19	0.28 ± 0.21
10u	GraphCast-Improved	**0.47 ± 0.08**	**0.44 ± 0.10**	**0.75 ± 0.10**	**0.29 ± 0.11**	**0.35 ± 0.26**	0.20 ± 0.21	**0.35 ± 0.20**	**0.41 ± 0.25**
10u	FourCastNet	0.18 ± 0.06	0.19 ± 0.07	0.13 ± 0.10	0.16 ± 0.09	0.22 ± 0.21	**0.22 ± 0.18**	0.22 ± 0.20	0.18 ± 0.20
10v	GraphCast-Baseline	0.27 ± 0.07	0.31 ± 0.08	0.41 ± 0.10	0.16 ± 0.10	0.10 ± 0.12	0.12 ± 0.13	0.22 ± 0.18	0.18 ± 0.21
10v	GraphCast-Improved	**0.37 ± 0.08**	**0.39 ± 0.09**	**0.57 ± 0.13**	**0.25 ± 0.11**	**0.17 ± 0.21**	**0.24 ± 0.24**	**0.24 ± 0.20**	**0.32 ± 0.25**
10v	FourCastNet	0.03 ± 0.04	0.04 ± 0.05	0.08 ± 0.08	0.02 ± 0.03	0.09 ± 0.13	0.14 ± 0.18	0.10 ± 0.15	0.07 ± 0.12
u_850	GraphCast-Baseline	0.26 ± 0.09	0.19 ± 0.09	0.47 ± 0.22	0.21 ± 0.11	0.21 ± 0.20	0.07 ± 0.14	0.23 ± 0.21	0.28 ± 0.22
u_850	GraphCast-Improved	**0.44 ± 0.09**	**0.41 ± 0.11**	**0.70 ± 0.10**	**0.31 ± 0.12**	**0.42 ± 0.27**	0.20 ± 0.21	**0.32 ± 0.23**	**0.43 ± 0.24**
u_850	FourCastNet	0.18 ± 0.07	0.21 ± 0.08	0.13 ± 0.10	0.15 ± 0.10	0.22 ± 0.20	**0.27 ± 0.21**	0.25 ± 0.21	0.21 ± 0.24
v_850	GraphCast-Baseline	0.17 ± 0.06	0.21 ± 0.08	0.23 ± 0.11	0.12 ± 0.08	0.10 ± 0.13	0.13 ± 0.14	**0.24 ± 0.18**	0.20 ± 0.20
v_850	GraphCast-Improved	**0.29 ± 0.08**	**0.34 ± 0.10**	**0.39 ± 0.10**	**0.19 ± 0.10**	**0.18 ± 0.22**	**0.28 ± 0.26**	0.22 ± 0.20	**0.29 ± 0.26**
v_850	FourCastNet	0.03 ± 0.04	0.05 ± 0.05	0.06 ± 0.06	0.04 ± 0.04	0.09 ± 0.14	0.12 ± 0.15	0.09 ± 0.13	0.05 ± 0.08

Table D.6: ACC metric scores (mean ± standard deviation) for 7 days forecast lead time. For `GraphCast-Baseline` and `GraphCast-Improved` $n = 81$, and for `FourCastNet` $n = 27$. Higher values are better.

Metric	Variable	Lead time	global	n.hem	tropics	s.hem	europe	n.amer	e.asia	austnz
RMSE	all	1 day	**5.34e-15 (d=5.6)**	**5.35e-15 (d=4.3)**	**5.34e-15 (d=5.7)**	**5.35e-15 (d=4.1)**	**6.45e-15 (d=1.8)**	**6.96e-15 (d=1.5)**	**5.34e-15 (d=3.4)**	**8.10e-13 (d=1.1)**
RMSE	all	3 days	**5.35e-15 (d=3.5)**	**5.36e-15 (d=3.1)**	**5.35e-15 (d=2.9)**	**5.56e-15 (d=2.0)**	**1.09e-14 (d=1.3)**	**3.03e-13 (d=1.0)**	**5.56e-15 (d=2.3)**	**2.21e-09 (d=0.8)**
RMSE	all	7 days	**5.34e-15 (d=2.7)**	**5.36e-15 (d=2.9)**	**5.36e-15 (d=1.8)**	**4.29e-13 (d=1.2)**	**5.81e-14 (d=1.2)**	**1.97e-10 (d=0.8)**	**5.36e-15 (d=1.6)**	**7.52e-07 (d=0.6)**
RMSE	z_500	1 day	**5.34e-15 (d=3.1)**	**5.55e-15 (d=2.3)**	**5.35e-15 (d=1.8)**	**5.35e-15 (d=1.8)**	**6.95e-15 (d=1.7)**	**5.36e-15 (d=1.6)**	**5.77e-15 (d=1.7)**	**4.83e-10 (d=0.9)**
RMSE	z_500	3 days	**5.35e-15 (d=2.4)**	**5.36e-15 (d=1.9)**	**5.36e-15 (d=1.7)**	**5.35e-15 (d=2.0)**	**6.36e-14 (d=1.4)**	**1.72e-12 (d=1.0)**	**5.77e-15 (d=1.9)**	**1.28e-11 (d=1.0)**
RMSE	z_500	7 days	**5.35e-15 (d=2.1)**	**5.36e-15 (d=1.9)**	**5.78e-15 (d=1.5)**	**1.49e-14 (d=1.5)**	**1.09e-14 (d=1.5)**	**2.13e-10 (d=0.9)**	**8.15e-15 (d=1.6)**	**2.63e-13 (d=0.9)**
RMSE	t_850	1 day	**5.35e-15 (d=3.1)**	**5.36e-15 (d=2.4)**	**5.36e-15 (d=3.0)**	**5.35e-15 (d=2.8)**	**5.56e-15 (d=1.9)**	**5.36e-15 (d=2.0)**	**5.35e-15 (d=2.2)**	**5.35e-15 (d=2.0)**
RMSE	t_850	3 days	**5.36e-15 (d=2.4)**	**5.36e-15 (d=1.9)**	**5.36e-15 (d=2.4)**	**5.35e-15 (d=2.5)**	**7.89e-14 (d=1.1)**	**3.48e-11 (d=0.8)**	**7.09e-15 (d=1.5)**	**9.48e-15 (d=1.6)**
RMSE	t_850	7 days	**5.36e-15 (d=2.5)**	**5.36e-15 (d=2.4)**	**5.36e-15 (d=2.0)**	**5.36e-15 (d=2.0)**	**5.36e-15 (d=1.7)**	**9.08e-12 (d=0.9)**	**5.67e-15 (d=1.5)**	**5.04e-14 (d=1.1)**
RMSE	t2m	1 day	**5.36e-15 (d=3.3)**	**5.36e-15 (d=2.9)**	**5.36e-15 (d=3.0)**	**5.36e-15 (d=2.7)**	**5.36e-15 (d=1.4)**	**5.36e-15 (d=1.4)**	**5.36e-15 (d=1.5)**	**5.78e-15 (d=2.0)**
RMSE	t2m	3 days	**5.36e-15 (d=2.4)**	**5.36e-15 (d=2.1)**	**5.36e-15 (d=2.6)**	**5.36e-15 (d=2.0)**	**6.83e-15 (d=1.2)**	**7.85e-15 (d=1.0)**	**5.36e-15 (d=1.4)**	**1.69e-14 (d=1.5)**
RMSE	t2m	7 days	**5.36e-15 (d=2.3)**	**5.36e-15 (d=2.5)**	**5.36e-15 (d=1.8)**	**6.34e-15 (d=1.5)**	**9.37e-15 (d=1.5)**	**5.56e-15 (d=1.0)**	**5.36e-15 (d=1.4)**	**5.37e-10 (d=0.8)**
RMSE	msl	1 day	**5.34e-15 (d=3.4)**	**5.35e-15 (d=3.8)**	**4.72e-13 (d=1.2)**	**5.55e-15 (d=1.4)**	**5.56e-15 (d=2.2)**	**1.13e-14 (d=1.7)**	**7.88e-14 (d=1.3)**	**3.65e-11 (d=1.0)**
RMSE	msl	3 days	**1.91e-14 (d=1.7)**	**5.33e-15 (d=2.8)**	**4.54e-12 (d=1.0)**	3.61e-01	**4.87e-12 (d=1.0)**	**5.87e-11 (d=0.9)**	**7.66e-07 (d=0.6)**	**1.24e-04 (d=0.5)**
RMSE	msl	7 days	**1.25e-04 (d=0.5)**	**2.00e-12 (d=1.1)**	**2.15e-11 (d=1.0)**	**4.05e-02** (d=-0.3)	1.81e-01	**6.69e-04 (d=0.4)**	**2.52e-02 (d=0.3)**	**1.43e-03 (d=0.5)**
RMSE	10u	1 day	**5.32e-15 (d=6.0)**	**5.35e-15 (d=5.2)**	**5.33e-15 (d=3.7)**	**5.32e-15 (d=3.8)**	**5.35e-15 (d=3.2)**	**5.35e-15 (d=2.7)**	**5.35e-15 (d=2.7)**	**5.56e-15 (d=1.8)**
RMSE	10u	3 days	**5.32e-15 (d=3.1)**	**5.34e-15 (d=2.4)**	**5.35e-15 (d=1.9)**	**5.31e-15 (d=2.5)**	**5.25e-14 (d=1.4)**	**3.42e-13 (d=1.2)**	**1.37e-11 (d=1.0)**	**2.25e-12 (d=1.1)**
RMSE	10u	7 days	**5.34e-15 (d=2.6)**	**5.35e-15 (d=2.3)**	**5.36e-15 (d=1.8)**	**5.52e-15 (d=1.8)**	**6.38e-13 (d=1.3)**	**1.44e-09 (d=0.9)**	**7.48e-09 (d=0.8)**	**9.00e-09 (d=0.8)**
RMSE	10v	1 day	**5.30e-15 (d=9.2)**	**5.34e-15 (d=6.9)**	**5.34e-15 (d=3.2)**	**5.35e-15 (d=4.7)**	**5.36e-15 (d=3.3)**	**5.36e-15 (d=3.3)**	**5.35e-15 (d=3.1)**	**5.36e-15 (d=1.7)**
RMSE	10v	3 days	**5.30e-15 (d=2.9)**	**1.74e-14 (d=2.0)**	**2.54e-13 (d=1.2)**	**5.33e-15 (d=2.2)**	**3.01e-07 (d=0.6)**	**6.52e-06 (d=0.5)**	**2.35e-10 (d=0.9)**	**1.51e-09 (d=0.9)**
RMSE	10v	7 days	**6.68e-15 (d=1.6)**	**1.12e-10 (d=0.9)**	**8.37e-15 (d=1.8)**	**1.43e-14 (d=1.5)**	**3.02e-02 (d=0.3)**	**2.67e-02 (d=0.3)**	**9.13e-04 (d=0.4)**	**3.45e-07 (d=0.7)**
RMSE	u_850	1 day	**5.32e-15 (d=7.0)**	**5.34e-15 (d=6.0)**	**5.30e-15 (d=2.7)**	**5.33e-15 (d=3.9)**	**5.35e-15 (d=3.0)**	**5.36e-15 (d=2.8)**	**5.34e-15 (d=2.8)**	**5.36e-15 (d=2.1)**
RMSE	u_850	3 days	**5.34e-15 (d=3.3)**	**5.35e-15 (d=2.6)**	**5.34e-15 (d=1.5)**	**5.34e-15 (d=2.4)**	**4.05e-14 (d=1.5)**	**1.31e-12 (d=1.2)**	**9.26e-14 (d=1.4)**	**2.97e-13 (d=1.2)**
RMSE	u_850	7 days	**5.35e-15 (d=2.4)**	**5.33e-15 (d=2.1)**	**5.36e-15 (d=1.5)**	**5.54e-15 (d=2.0)**	**3.79e-13 (d=1.4)**	**2.57e-11 (d=1.0)**	**9.91e-11 (d=1.0)**	**2.92e-11 (d=1.0)**
RMSE	v_850	1 day	**5.31e-15 (d=7.9)**	**5.34e-15 (d=6.4)**	**5.34e-15 (d=2.4)**	**5.33e-15 (d=4.9)**	**5.35e-15 (d=3.7)**	**5.36e-15 (d=3.2)**	**5.35e-15 (d=2.9)**	**5.35e-15 (d=1.6)**
RMSE	v_850	3 days	**5.33e-15 (d=3.2)**	**1.04e-14 (d=2.3)**	**9.18e-15 (d=1.4)**	**7.83e-15 (d=2.2)**	**8.67e-10 (d=0.9)**	**6.46e-08 (d=0.7)**	**1.37e-11 (d=1.0)**	**1.51e-09 (d=0.8)**
RMSE	v_850	7 days	**5.35e-15 (d=1.9)**	**1.95e-13 (d=1.4)**	**4.33e-14 (d=1.4)**	**9.46e-15 (d=1.8)**	**6.01e-03 (d=0.3)**	**2.38e-04 (d=0.5)**	**2.21e-04 (d=0.4)**	**4.63e-07 (d=0.6)**

Table D.7: Wilcoxon signed-rank test p-values for pairwise comparison between the baseline and improved models. Bold values indicate statistically significant differences ($p < 0.05$). The corresponding effect sizes (Cohen's d) are reported to quantify the magnitude of the difference. Bold values indicate that the improved model had better values for the metric.

Metric	Variable	Lead time	global	n.hem	tropics	s.hem	europe	n.amer	e.asia	austnz
ACC	all	1 day	**7.04e-11** (d=1.0)	**1.10e-14 (d=-2.2)**	**4.35e-15** (d=4.2)	**8.44e-15** (d=1.9)	**2.09e-12** (d=0.9)	**1.82e-13 (d=-1.2)**	**5.26e-15** (d=2.0)	**5.32e-15** (d=3.1)
ACC	all	3 days	6.66e-02	**5.30e-15 (d=-1.9)**	**4.84e-15** (d=3.2)	**3.49e-11** (d=1.0)	**4.78e-08** (d=0.7)	**2.78e-04 (d=-0.4)**	**5.32e-15** (d=2.0)	**5.34e-15** (d=3.1)
ACC	all	7 days	**6.56e-05** (d=0.5)	**5.52e-15 (d=-1.8)**	**5.11e-15** (d=2.8)	**4.07e-13** (d=1.4)	**3.40e-06** (d=0.5)	6.08e-02	**2.76e-14** (d=1.3)	**5.34e-15** (d=3.3)
ACC	z_500	1 day	**5.34e-15 (d=-2.8)**	**5.55e-15 (d=-1.7)**	**1.73e-03 (d=-0.4)**	**5.34e-15 (d=-2.1)**	**1.69e-14 (d=-1.3)**	**5.55e-15 (d=-1.4)**	**4.44e-14 (d=-1.3)**	**3.16e-08 (d=-0.6)**
ACC	z_500	3 days	**5.36e-15 (d=-2.5)**	**2.53e-14 (d=-1.6)**	**2.46e-04 (d=-0.5)**	**2.72e-14 (d=-1.6)**	**6.31e-10 (d=-0.9)**	**1.03e-04 (d=-0.5)**	**2.32e-11 (d=-1.1)**	**1.81e-06 (d=-0.6)**
ACC	z_500	7 days	**5.62e-10 (d=-0.9)**	**3.77e-08 (d=-0.7)**	**1.10e-02 (d=-0.3)**	**1.01e-06 (d=-0.6)**	**1.82e-06 (d=-0.6)**	2.56e-01	**9.05e-04 (d=-0.4)**	**3.10e-03 (d=-0.4)**
ACC	t_850	1 day	**5.31e-15 (d=-2.6)**	**5.34e-15 (d=-2.0)**	**5.35e-15 (d=-2.3)**	**5.31e-15 (d=-2.4)**	**5.34e-15 (d=-1.3)**	**5.35e-15 (d=-1.6)**	**6.44e-15 (d=-1.6)**	**5.95e-15 (d=-1.1)**
ACC	t_850	3 days	**5.35e-15 (d=-4.0)**	**5.36e-15 (d=-2.7)**	**5.35e-15 (d=-2.1)**	**5.35e-15 (d=-2.5)**	**1.33e-12 (d=-1.1)**	**3.84e-04 (d=-0.4)**	**2.25e-13 (d=-1.3)**	**4.52e-13 (d=-1.0)**
ACC	t_850	7 days	**5.36e-15 (d=-2.6)**	**1.49e-14 (d=-1.7)**	**8.38e-15 (d=-1.6)**	**1.28e-13 (d=-1.4)**	**2.45e-09 (d=-0.8)**	8.65e-01	**1.45e-09 (d=-0.8)**	**2.80e-06 (d=-0.6)**
ACC	t2m	1 day	**5.33e-15 (d=-2.4)**	**5.31e-15 (d=-1.9)**	**5.35e-15 (d=-2.6)**	**5.34e-15 (d=-1.8)**	**7.83e-15 (d=-1.1)**	**4.75e-14 (d=-1.2)**	**5.36e-15 (d=-1.3)**	**5.66e-15 (d=-1.3)**
ACC	t2m	3 days	**5.35e-15 (d=-2.7)**	**5.35e-15 (d=-2.2)**	**5.36e-15 (d=-2.8)**	**5.36e-15 (d=-2.2)**	**2.45e-13 (d=-1.2)**	**7.55e-06 (d=-0.6)**	**8.46e-15 (d=-1.8)**	**5.31e-14 (d=-1.3)**
ACC	t2m	7 days	**5.36e-15 (d=-2.4)**	**5.99e-15 (d=-2.1)**	**5.36e-15 (d=-2.5)**	**9.84e-15 (d=-1.7)**	**4.88e-12 (d=-1.2)**	**5.97e-05 (d=-0.5)**	**7.49e-15 (d=-1.5)**	**1.18e-11 (d=-1.1)**
ACC	msl	1 day	**5.34e-15 (d=-3.1)**	**5.35e-15 (d=-2.7)**	**5.54e-15 (d=-2.1)**	**5.35e-15 (d=-1.9)**	**6.22e-15 (d=-1.6)**	**1.03e-14 (d=-1.7)**	**1.61e-13 (d=-1.2)**	**4.79e-12 (d=-1.1)**
ACC	msl	3 days	**5.35e-15 (d=-2.4)**	**5.36e-15 (d=-2.8)**	**2.16e-14 (d=-1.7)**	**3.02e-09 (d=-0.8)**	**1.54e-06 (d=-0.6)**	**6.53e-05 (d=-0.5)**	**7.81e-05 (d=-0.5)**	**1.63e-05 (d=-0.5)**
ACC	msl	7 days	**1.24e-11 (d=-1.1)**	**9.95e-14 (d=-1.4)**	**1.32e-13 (d=-1.5)**	**1.25e-04 (d=-0.4)**	9.61e-01	4.94e-01	8.56e-01	**5.17e-04 (d=-0.4)**
ACC	10u	1 day	**5.29e-15 (d=-4.8)**	**5.32e-15 (d=-4.0)**	**4.80e-15 (d=-2.3)**	**5.31e-15 (d=-4.0)**	**5.35e-15 (d=-2.1)**	**5.35e-15 (d=-2.3)**	**5.35e-15 (d=-2.2)**	**6.08e-15 (d=-1.3)**
ACC	10u	3 days	**5.35e-15 (d=-3.3)**	**5.35e-15 (d=-2.5)**	**8.60e-15 (d=-1.9)**	**8.21e-15 (d=-2.0)**	**4.48e-12 (d=-1.2)**	**6.26e-12 (d=-1.1)**	**1.15e-06 (d=-0.6)**	**3.30e-09 (d=-0.8)**
ACC	10u	7 days	**5.34e-15 (d=-2.7)**	**5.36e-15 (d=-2.4)**	**9.89e-15 (d=-1.3)**	**8.03e-10 (d=-0.9)**	**1.04e-05 (d=-0.5)**	**5.52e-04 (d=-0.4)**	**1.62e-04 (d=-0.5)**	**4.60e-05 (d=-0.5)**
ACC	10v	1 day	**5.27e-15 (d=-6.4)**	**5.31e-15 (d=-6.0)**	**5.21e-15 (d=-3.1)**	**5.33e-15 (d=-4.3)**	**5.36e-15 (d=-2.7)**	**5.36e-15 (d=-2.9)**	**5.35e-15 (d=-2.4)**	**5.34e-15 (d=-1.2)**
ACC	10v	3 days	**5.35e-15 (d=-3.4)**	**5.77e-15 (d=-2.6)**	**8.46e-12 (d=-1.2)**	**5.99e-15 (d=-2.1)**	**1.16e-10 (d=-1.0)**	**1.94e-07 (d=-0.7)**	**4.75e-12 (d=-1.1)**	**8.97e-08 (d=-0.7)**
ACC	10v	7 days	**1.42e-13 (d=-1.4)**	**1.22e-11 (d=-1.1)**	**3.21e-12 (d=-1.2)**	**2.80e-07 (d=-0.7)**	5.35e-02	**3.27e-05 (d=-0.5)**	4.42e-01	**2.88e-04 (d=-0.4)**
ACC	u_850	1 day	**5.23e-15 (d=-5.2)**	**5.31e-15 (d=-4.1)**	**5.02e-15 (d=-1.7)**	**5.27e-15 (d=-4.2)**	**5.34e-15 (d=-1.7)**	**5.35e-15 (d=-2.2)**	**5.35e-15 (d=-2.1)**	**7.81e-15 (d=-1.3)**
ACC	u_850	3 days	**5.34e-15 (d=-3.4)**	**5.36e-15 (d=-2.4)**	**6.51e-15 (d=-1.8)**	**5.55e-15 (d=-2.2)**	**1.01e-12 (d=-1.3)**	**1.04e-11 (d=-1.1)**	**1.40e-08 (d=-0.8)**	**1.97e-10 (d=-0.9)**
ACC	u_850	7 days	**5.35e-15 (d=-2.2)**	**6.70e-15 (d=-2.0)**	**1.02e-14 (d=-1.3)**	**1.16e-08 (d=-0.8)**	**6.55e-08 (d=-0.7)**	**4.50e-05 (d=-0.5)**	**2.08e-04 (d=-0.4)**	**7.28e-06 (d=-0.6)**
ACC	v_850	1 day	**5.29e-15 (d=-6.2)**	**5.32e-15 (d=-5.9)**	**5.31e-15 (d=-2.6)**	**5.33e-15 (d=-4.4)**	**5.36e-15 (d=-2.6)**	**5.36e-15 (d=-2.6)**	**5.36e-15 (d=-2.4)**	**5.34e-15 (d=-1.2)**
ACC	v_850	3 days	**5.35e-15 (d=-3.5)**	**5.99e-15 (d=-2.9)**	**6.96e-15 (d=-1.9)**	**1.38e-14 (d=-1.9)**	**4.55e-11 (d=-1.0)**	**3.21e-08 (d=-0.8)**	**2.84e-12 (d=-1.1)**	**9.21e-07 (d=-0.6)**
ACC	v_850	7 days	**2.45e-13 (d=-1.4)**	**2.49e-13 (d=-1.3)**	**1.32e-13 (d=-1.4)**	**1.98e-06 (d=-0.6)**	5.59e-02	**8.53e-06 (d=-0.6)**	5.99e-01	**2.78e-03 (d=-0.3)**

Table D.8: Wilcoxon signed-rank test p-values for pairwise comparison between the baseline and improved models. Bold values indicate statistically significant differences ($p < 0.05$). The corresponding effect sizes (Cohen's d) are reported to quantify the magnitude of the difference. Bold values indicate that the improved model had better values for the metric.