TADPOLE-2D: FROM PDE FOUNDATION MODEL TO GLOBAL WEATHER FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

The development of AI-based weather forecasting models generally demands significant computational resources and extensive data storage. Recent advances in foundation models for partial differential equations (PDEs) have enabled efficient synthetic datasets and storage-free training pipelines, which may address these challenges. In this work, we present Tadpole-2D, a versatile foundation model for two-dimensional (2D) PDEs. It is pretrained to reconstruct 2D slices from diverse three-dimensional PDE data generated online, and its downstream performance is verified on a challenging PDE dynamics learning task. We extend the model to global weather prediction with adaptations enabling it to process atmospheric data on the Hierarchical Equal Area isoLatitude Pixelization (HEALPix) grid. We choose HEALPix for its superior numerical properties over equiangular grids used in most flagship weather systems. In short-range weather forecasting, the resulting model outperforms state-of-the-art PDE foundation models of comparable size. To our knowledge, ours is the first work in which a PDE foundation model is demonstrated to transfer successfully to global weather forecasting.

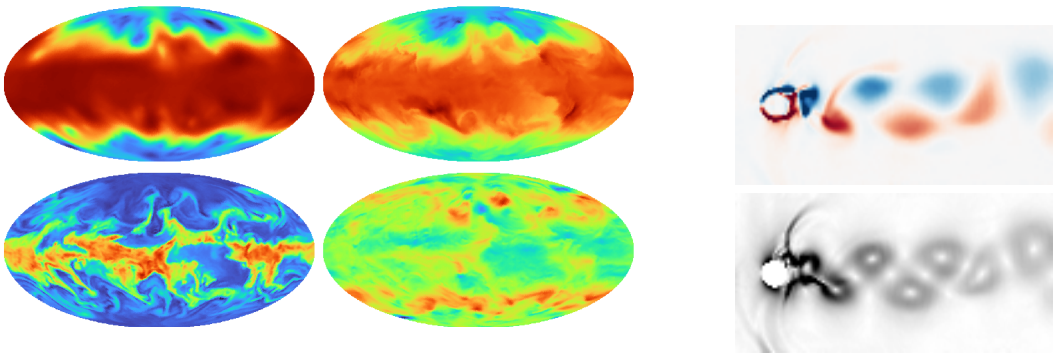
1 INTRODUCTION

Accurate weather forecasting supports agriculture, energy trading, and the early warning of disasters and extreme events. Numerical weather prediction (NWP) plays a critical role in reliable weather prediction (Bauer et al., 2015). Nevertheless, NWP typically demands substantial computational resources due to the high cost of spatiotemporal discretization of atmospheric equations. Recently, AI-based weather forecasting systems have shown considerable potential to accelerate or even replace NWP (Rasp et al., 2024; Bi et al., 2023; Lam et al., 2023; Nguyen et al., 2023; Bodnar et al., 2025; Schmude et al., 2024; Lang et al., 2024). However, training these AI models is costly and necessitates large volumes of data. The generation and management of these data present significant challenges for the development of advanced AI-based weather models.

Atmospheric dynamics is governed by partial differential equations (PDEs) that describe how state variables such as wind velocity, temperature, pressure, and humidity change over space and time. Consequently, AI-based weather prediction shares significant similarities with AI-based PDE modelling. Recently, foundation models have gained prominence within the PDE research community (Herde et al., 2024; Hao et al., 2024; McCabe et al., 2026). These models are typically trained on large-scale datasets and can be adapted to a range of downstream PDE tasks. While PDE foundation models also require substantial amounts of simulation data, generating and processing these data costs considerably less than weather data. Moreover, recent studies indicate that PDE foundation models trained on simple synthetic PDEs with fast online simulations can generalize to more complex simulations (Liu et al., 2026). Therefore, adapting PDE foundation models to weather forecasting is a promising approach to reduce the costs and improve the performance of upcoming AI-based weather prediction models.

In this work, we present Tadpole-2D, a two-dimensional PDE foundation model that can be efficiently fine-tuned for weather systems. Building on previous research, we employ an online learning method (Meyer et al., 2023; Liu et al., 2026) with a GPU-based pseudospectral solver (Liu et al., 2025) to generate diverse PDE data for pretraining. This approach enables training on several terabytes of samples while the simulator runs in parallel, using only two workstation-grade GPUs and no local data storage. We fine-tune Tadpole-2D for short-range weather forecasting (up to 24

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



Weather Forecast on the HEALPix Grid

Transonic Dynamics

Figure 1: Left: We adapt our model for downstream weather forecasting on the HEALPix grid. Shown here are `z500`, `t850`, `q700`, and `u850` fields from our model's one-step forecast. Right: We verify its ability to generalise to an unseen PDE setting - transonic flow around a cylinder. Vorticity and Schlieren fields computed from our model's flow predictions are shown.

hours) using the ERA5 dataset (Hersbach et al., 2020). The model operates on the Hierarchical Equal Area isoLatitude Pixelization (HEALPix) grid (Górski et al., 2005), which discretizes the sphere into cells of equal area. This grid provides a superior alternative to the standard equiangular latitude–longitude grid, which oversamples the poles. We implement a HEALPix interface that enables our model to operate on the grid's hierarchical face layout and handles the complex treatment of face neighbours, following Karlbauer et al. (2024). In our experiments, the pretrained model consistently outperforms its non-pretrained counterpart (trained from scratch on weather prediction) across all short-range weather forecasting lead times, given identical fine-tuning data budgets and update steps. Furthermore, our model achieves the best performance among similarly sized PDE foundation models while requiring the fewest trainable parameters during fine-tuning. Finally, we discuss methods from flagship weather models that could be transferred to PDE foundation models to improve performance further. To the authors' knowledge, Tadpole-2D is the first PDE foundation model adapted to global weather forecasting.

The main contributions of this work are as follows:

- We introduce Tadpole-2D, a two-dimensional PDE foundation model pretrained on online PDE simulations with no training data storage requirements.
- We propose using PDE foundation models for downstream weather forecasting tasks and demonstrate their adaptation to modelling on the HEALPix spherical grid.
- We demonstrate the versatility of our model by validating its fine-tuning performance on both weather forecasting and PDE dynamics emulation.

2 Related Work

AI-based weather forecasting has rapidly advanced, especially over the past five years. Pangu-Weather and GraphCast outperform the deterministic high-resolution forecast of the European Centre for Medium-Range Weather Forecasts (ECMWF) for most of the variables and lead times they evaluate (Bi et al., 2023; Lam et al., 2023). GraphCast produces a ten-day global forecast in under a minute. Other notable approaches include probabilistic and hybrid models (Price et al., 2025; Kochkov et al., 2024) and weather foundation models (Nguyen et al., 2023; Bodnar et al., 2025; Schmude et al., 2024). WeatherBench 2 (Rasp et al., 2024) provides a well-established benchmark for machine-learning weather prediction models.

Most of these models use the equiangular latitude-longitude grid, which oversamples the poles. The HEALPix grid offers a superior alternative as it divides the sphere into cells of equal area (Górski et al., 2005). Recent work has begun adopting HEALPix for AI-based weather prediction. Recent works using this grid include DLWP-HPX (Karlbauer et al., 2024), HEAL-ViT (Ramavajjala, 2024),

HEAL-SWIN (Carlsson et al., 2024), DeepSphere (Perraudin et al., 2019), and PEAR (Linander et al., 2025). Notably, Destination Earth - an ECMWF initiative aimed at creating earth digital twins for climate change and extreme weather prediction - uses HEALPix as its standard grid (Sandu, 2024).

Most weather models are trained on roughly four decades of ERA5 data. The observational record grows only as fast as time passes. Moreover, the datasets are large; hourly ERA5 occupies petabytes. PDE foundation models sidestep both limits because their pretraining data are simulated rather than observed. Poseidon pretrains on compressible Euler and incompressible Navier–Stokes solutions (Herde et al., 2024), DPOT on twelve datasets drawn from established PDE benchmarks (Hao et al., 2024), PDEformer-2 on a mix of eight equation families (Ye et al., 2025), MORPH on six benchmark datasets (Rautela et al., 2025), and Walrus on nineteen scenarios from The Well and FlowBench, ranging from plasma physics to astrophysics (McCabe et al., 2026). All these models use further simulated PDE data for downstream evaluations.

3 METHODOLOGY

In this section, we discuss the design of Tadpole-2D, focusing on its pretraining on simple PDE data, its fine-tuning for weather prediction and a verification task on transonic flow around a cylinder.

3.1 PRETRAINING

3.1.1 TRAINING OBJECTIVE

For pretraining, our design decisions and training framework follow Liu et al. (2026)'s central idea of splitting the learning task into spatial and temporal regimes. Pretraining is restricted to spatial reconstruction, which enables the model to learn efficient latent representations of fields from diverse partial differential equation (PDE) systems. Modelling time evolution is deferred to downstream learning. Due to this decoupling of spatial and temporal features, Tadpole-2D does not require physical or simulation parameter embeddings during pretraining, since these parameters typically govern the PDE's temporal dynamics. Furthermore, reconstruction pretraining allows each variable to be treated as an independent scalar field. In practice, the encoder and decoder process each data channel independently, and the channels are combined only during fine-tuning for downstream tasks. We achieve this by collapsing the channel dimension into the batch dimension, so an input of shape $[B, C, H, W]$ is treated during pretraining as $[B \cdot C, 1, H, W]$.

We perform spatial reconstruction pretraining by training Tadpole-2D as a variational autoencoder (VAE) (Kingma & Welling, 2014). The encoder $\mathcal{E}$ maps a scalar input field u to a latent distribution $q_{\mathcal{E}}(\boldsymbol{z} \mid u)$, while the decoder $\mathcal{D}$ reconstructs the field from a latent variable $\boldsymbol{z}$ sampled from this distribution. The training objective consists of a reconstruction likelihood term and a Kullback–Leibler (KL) divergence term, weighted by λ_{KL}, that encourages the latent distribution to approach a standard normal prior $p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$:

$$\mathcal{L}_{\mathrm{VAE}} = \mathbb{E}_{q_{\mathcal{E}}(\boldsymbol{z}|u)} \left[-\log p_{\mathcal{D}}(u \mid \boldsymbol{z}) \right] + \lambda_{\mathrm{KL}} \, D_{\mathrm{KL}}(q_{\mathcal{E}}(\boldsymbol{z} \mid u) \,\|\, p(\boldsymbol{z})) \,. \tag{1}$$

Here, λ_{KL} is set to 10^{-6}. Additional training details and hyperparameters are provided in Appendix A.4.1. For the network backbone, we use the P3D architecture (Holzschuh et al., 2026), a state-of-the-art transformer designed for 3D PDE emulation. To adapt the model to two-dimensional data, we replace the three-dimensional convolutions with two-dimensional convolutions. A full specification of the neural network is provided in Appendix A.2.

3.1.2 ONLINE DATA GENERATION

The pretraining dataset comprises seven families of PDEs: diffusion, hyperdiffusion, viscous Burgers, a multidimensional Korteweg-de Vries system, Kuramoto-Sivashinsky, Fisher-KPP, and Swift-Hohenberg. This selection encompasses diffusive, dispersive, advective, chaotic, reaction-diffusion, and pattern-forming phenomena. For each PDE, fields are simulated at resolutions of 64^3, 128^3, 256^3, and 384^3 using the Fourier-spectral solver `TorchFSM` (Liu et al., 2025). The model is trained on two-dimensional slices extracted from these three-dimensional simulations. Utilising slices from 3D systems for 2D model training exposes the model to a broader range of spatial structures than purely 2D simulations typically provide.

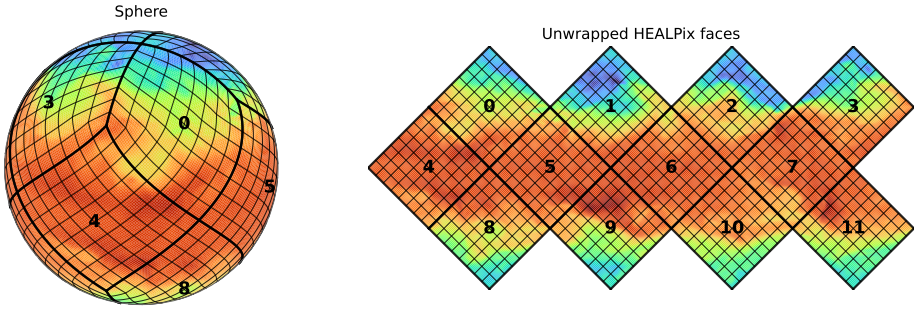


Figure 2: An illustration of the HEALPix grid showing our model's one-step forecast of the `t850` field. The numbers mark indices of the 12 base faces.

Instead of relying on a precomputed local dataset, we employ an online learning approach. `TorchFSM` generates data in real time during training and updates a buffer from which the training process retrieves samples. Each generated 3D state is randomly sliced and cropped to a 64×64 domain before retrieval from the buffer. The entire pretraining process required 16.6 days on two NVIDIA RTX A5000 GPUs, each with 24 GB of memory. One GPU was dedicated to model training and the other to running `TorchFSM`. This online pipeline yielded a total of 2.2×10^5 simulation runs and 1.16×10^9 2D slices. Details of the PDE configurations, simulation parameters, and dataset statistics observed during pretraining are provided in Appendix A.6.1.

3.2 Fine-tuning for Weather Forecasting

3.2.1 Dynamics Fine-tuning

We transform the pretrained Tadpole-2D model from an autoencoder to a dynamics predictor for weather forecasting using the Tadpole-DFT method (Liu et al., 2026). In this approach, a subnetwork is inserted in the bottleneck between the encoder and decoder to aggregate the input variables. As discussed in Section 3.1.1, the encoder and decoder process each variable independently. The variables are unfolded into the feature dimension in the latent space to enable their interaction for learning dynamics via the subnetwork. Low-rank adaptation (LoRA) (Hu et al., 2022) is applied to both the encoder and decoder, enabling the backbone network to learn relevant features effectively. Additionally, skip connections are reintroduced between corresponding layers of the encoder and the decoder, allowing the model to capture dynamic transformations across multiple compression levels. The architecture of our subnetwork is given in Appendix A.3. Further details of the Tadpole-DFT method are discussed extensively in Liu et al. (2026).

3.2.2 HEALPix Backbone Interface

Equiangular latitude–longitude grids projected onto the Earth's spherical topology oversample the poles. The HEALPix grid overcomes this numerical limitation by dividing the sphere into 12 base faces, each further subdivided into $N_{\text{side}} \times N_{\text{side}}$ cells of equal area, with N_{side} cells along each edge and N_{side} a power of 2. Hence, each base face is a square grid that supports standard 2D convolution kernels. Figure 2 visualizes this grid. We use a grid with $N_{\text{side}} = 64$, designated as `HPX64`, containing a total of 49,152 cells and an angular resolution of $\approx 0.92^\circ$.

HEALPix-Aware Padding In Tadpole-2D, faces and state variables are processed as independent samples in the encoder and decoder. Although the state variables interact in the latent space, the interactions between HEALPix faces require careful treatment. The faces are not independent planar domains because the receptive fields of the convolutions require edge information from neighbouring faces. Their edges meet with rotations and reflections determined by the spherical topology. Following the face-neighbour construction used by DLWP-HPX (Karlbauer et al., 2024), we replace the native padding of every spatial convolution with a HEALPix-aware padding operation. Before each convolution, this operation copies boundary cells from the appropriate neighbouring face and rotates or reflects them into the receiving face's orientation. Only cell positions are rotated or reflected, so vector fields such as eastward and northward wind components retain their global orientation. Ambiguous diagonal cells at the corners of equatorial faces are formed by averaging the

two adjoining boundary values. Rebuilding this halo at every layer lets intermediate features, rather than only the raw input, cross face boundaries.

Global Positional Embeddings The backbone's windowed-attention blocks embed window-local cell indices, which assume uniform distances between cells. Moreover, because the backbone processes each HEALPix base face independently, no global positional information is available across faces. To inform the model of absolute positions and distances, we encode two geometric features of each grid cell: its normalized latitude, and the great-circle distance from its face's $(0, 0)$ corner. These coordinates are transferred from the native HEALPix grid to the latent grid through adaptive average pooling, and a shared 1×1 projection maps them to the token space. This projection is zero-initialized, preserving the initial pretrained map while allowing the dynamics network to learn a spherical position signal.

3.2.3 Dataset and Data Treatment

We use six-hourly ERA5 reanalysis fields (Hersbach et al., 2020) distributed through WeatherBench 2 (Rasp et al., 2024). Each source frame is stored on the $0.25°$ equiangular latitude-longitude grid with 721×1440 cells. We use 1979-2018 for training and 2019 for validation.

Choice of Variables We select the five atmospheric variables for which WeatherBench 2[1] publishes its headline scores: geopotential at 500 hPa (`z500`), temperature at 850 hPa (`t850`), specific humidity at 700 hPa (`q700`), and zonal and meridional wind at 850 hPa (`u850` and `v850`).

Data Standardization Each variable field is standardized to zero mean and unit variance using a mean and a population standard deviation accumulated over the training interval only. We prefer this over the min–max normalization to $[-1, 1]$ that the rest of our pipeline uses because a min–max map is set by the two most extreme cells ever observed in forty years of reanalysis. In contrast, standardization ties the scale to the bulk of the distribution.

Resolutions and Regridding We distinguish three grids: the dataset grid, the model grid, and the performance evaluation grid. Their respective resolutions are $0.25°$ latitude–longitude, $0.92°$ `HPX64`, and $1.5°$ latitude–longitude. During training, each dataset frame is downsampled to `HPX64` through sampled first-order conservative regridding, using weights proportional to the area of overlap between source and target cells. Neither WeatherBench 2 nor its successor, WeatherBench X, supports regridding between HEALPix and latitude–longitude grids. We use our own implementation, described in Appendix A.4.5.

3.2.4 Training Objective

The model is trained with a one-step mean squared error. Given the time-step size of the data, one model step produces a forecast 6 hours ahead, i.e., $\Delta t = 6\,\mathrm{h}$. Because HEALPix cells have equal area, a uniform mean over native cells is already a uniform spherical area average; unlike the equiangular latitude-longitude grid, no latitude weights are required.

Time-difference based Loss Weighting Because the five variables change over time at very different rates, a uniform mean over all channels would allow the fast wind channels to dominate the gradient, while the slow geopotential channel would contribute a term that is roughly 58 times smaller in variance. We therefore weight each channel by the inverse variance of its six-hour difference, following GraphCast (Lam et al., 2023). Appendix A.7.2 reports the per-channel statistics over the training split and the resulting weights.

Let σ_c denote the standard deviation of the six-hour difference of channel c in standardized units. We define the weights as

$$w_c = \frac{\sigma_c^{-2}}{\frac{1}{C}\sum_{c'=1}^{C} \sigma_{c'}^{-2}}, \tag{2}$$

[1] `https://sites.research.google/gr/weatherbench/deterministic-scores/`

which average to one, so the overall loss scale, and with it the usable learning rate, matches the uniformly weighted objective. For $C = 5$ and $N_{\text{side}} = 64$, the loss function is

$$\mathcal{L}_{\text{weather}} = \frac{1}{C}\sum_{c=1}^{C} w_c \frac{1}{12N_{\text{side}}^2}\sum_{f=1}^{12}\sum_{i=1}^{N_{\text{side}}}\sum_{j=1}^{N_{\text{side}}}\left(\hat{u}^{(c)}_{t+\Delta t,f,i,j} - u^{(c)}_{t+\Delta t,f,i,j}\right)^2, \tag{3}$$

where f indexes the base faces and (i, j) the cells within a face. The σ_c are estimated once from the training split and held fixed. At inference time, forecasts beyond six hours are generated autoregressively on the same HEALPix grid.

3.3 Learning PDE Dynamics

This task serves as a verification of the model's ability to learn downstream PDE dynamics. We fine-tune the model to emulate the time evolution of transonic cylinder flow. This system entails several features not seen during pretraining: non-periodic boundary conditions, transonic shocks, and a flow obstacle, making it a challenging task. We apply the Dynamics Fine-Tuning method explained in Section 3.2.1, using a LoRA rank of 64.

We use the dataset of Kohl et al. (2026), generated from simulations of compressible flow around a cylinder with the SU2 solver and the Spalart–Allmaras delayed detached-eddy simulation closure at Reynolds number 10^4. The fields lie on a 128×64 Cartesian grid and contain the velocity components (v_x, v_y), density, and pressure. The dataset spans Mach numbers from 0.50 to 0.90. We use Mach numbers 0.53–0.63 for training, 0.50–0.52 for validation, and 0.64–0.68 for testing. Model performance is evaluated in the developed-flow regime of each trajectory. The training objective is a one-step prediction MSE, uniform over the four channels and the grid, as defined in Appendix A.4.6. We normalize the input fields channel-wise to a minimum and maximum in the range $[-1, 1]$.

4 Experimental Results

4.1 Weather Forecasting

Model evaluation follows the WeatherBench 2 methodology for reporting headline deterministic scores (Rasp et al., 2024). Forecasts generated on the `HPX64` grid are conservatively remapped to the $1.5°$ (121×240) WeatherBench 2 evaluation grid, and errors are computed against the WeatherBench 2 ERA5 ground truth on this grid. The aggregated metrics include the latitude-weighted root-mean-square error (RMSE) and anomaly correlation coefficient (ACC) for the variables `z500`, `t850`, `q700`, `u850`, and `v850`. Appendix A.5 provides definitions for both metrics and the ACC climatology. The evaluation year is 2022, which is not included in any model's training or validation data. Each forecast is initialized from an ERA5 analysis at 00:00 or 12:00 UTC, yielding 730 test samples, and is rolled out autoregressively in six-hour time steps.

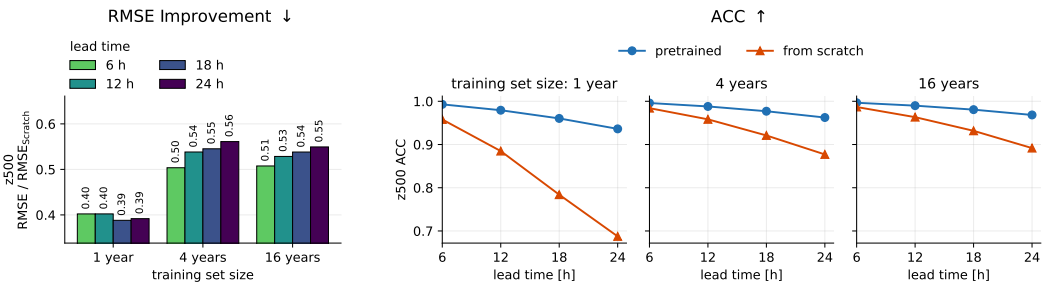


Figure 3: Left: Ratio of the `z500` forecast RMSE of the fine-tuned model to that of the model trained from scratch. Lower values indicate better performance by the fine-tuned model. Right: Anomaly correlation coefficient (ACC) of the `z500` forecasts from the fine-tuned and from-scratch models, where higher values indicate better accuracy. The figure also presents an ablation across different training set sizes. Appendix Figures 8 and 9 provide analogous comparisons for all five channels.

We first assess whether fine-tuning with the PDE-reconstruction-based pretraining pipeline offers advantages over training the model from scratch with randomly initialized weights. Fine-tuning employs 6.16 million trainable parameters, whereas training from scratch requires 23.63 million trainable parameters. Both variants were trained with identical hyperparameters, number of update steps, and data budgets, as detailed in Appendix A.4.3. During training, the models were exposed to one-step-ahead targets. We conducted four autoregressive evaluations, producing forecasts at lead times of 6, 12, 18, and 24 hours.

Figure 3 shows that pretraining improves the `z500` RMSE by 44% to 61% compared with the from-scratch model. The substantial RMSE gap at the smallest training dataset size indicates that pretraining improves sample efficiency. At longer lead times, errors accumulate as the autoregressive rollout length increases, and the RMSE gap between the pretrained and from-scratch models gradually narrows. In contrast, the ACC advantage of the fine-tuned model over the from-scratch model grows as lead time increases. ACC quantifies how accurately a forecast model predicts spatial anomaly patterns. This divergence suggests that the from-scratch model is more susceptible to autoregressive smoothing. For the pretrained model, the combination of a narrowing RMSE gap and an increasing ACC advantage indicates a better ability to learn spatial structures, attributable to reconstruction-based pretraining.

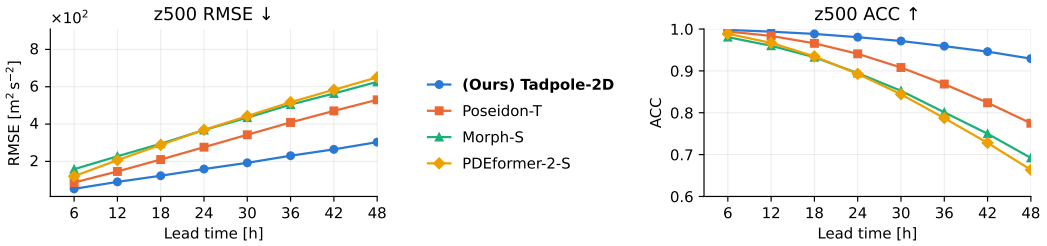


Figure 4: The performance of PDE foundation models of comparable size is evaluated in an autoregressive rollout over eight steps (48 hours) after fine-tuning on the one-step weather prediction task. Tadpole-2D performs best despite having the fewest trainable parameters. Figure 10 presents the comparison across all five channels.

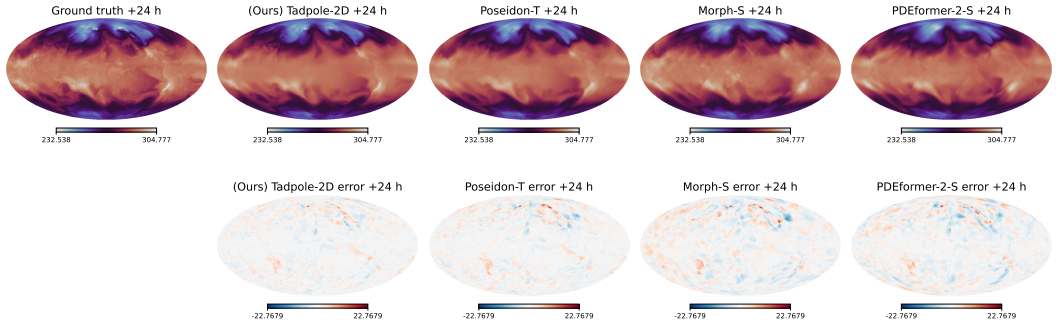


Figure 5: Temperature at 850 hPa (`t850`) forecast 24 hours ahead (four autoregressive six-hour steps), initialized from ERA5 at 00:00 UTC on 1 January 2022, shown in a Mollweide projection. Top row: ERA5 ground truth and each model's forecast (K). Bottom row: forecast error (K). Tadpole-2D's errors are visibly smaller and show no artefacts at the HEALPix base-face boundaries. MORPH-S shows square patch artefacts, and PDEformer-2-S shows the strongest seams along the face boundaries. Figures 11 to 15 in the appendix show the models' predictions of all five fields at more lead times.

Second, we benchmark Tadpole-2D against three state-of-the-art PDE foundation models of comparable size: Poseidon-T, MORPH-S, and PDEformer-2-S, which have 20.8 million, 32.8 million, and 27.7 million parameters, respectively. For Tadpole-2D, we employ a LoRA rank of 64 in this experiment, resulting in 8.4 million trainable parameters. All models are trained on the HEALPix grid for the same number of update steps and with equal batch sizes. During inference, each model is rolled

out autoregressively to generate forecasts up to 48 hours ahead with a 6-hour time step. Figure 5 compares the `t850` field of the models' 24-hour forecast. Tadpole-2D consistently outperforms all tested models, achieving better RMSE and ACC and exhibiting slower error growth and ACC decay, as shown in Figure 4. Compared with Poseidon-T, the second-best-performing model, Tadpole-2D achieves RMSE reductions of 10.5–42.7%, depending on the field, with the largest improvement observed for `z500`. Together, our pretraining pipeline and our HEALPix adaptation narrow the gap between PDE foundation models and dedicated weather models.

Figure 6 presents a comparison of Tadpole-2D with three state-of-the-art AI weather forecasting systems: GraphCast (Lam et al., 2023), Aurora (Bodnar et al., 2025), and Pangu-Weather (Bi et al., 2023). The reported scores for these systems were obtained from WeatherBench 2. Despite being trained on a lower-resolution grid and modelling fewer variables, the proposed model achieves a `q700` ACC within 0.03 of the baseline at the six-hour lead time. However, its predictive skill declines more rapidly at longer lead times, which we tentatively attribute to the use of one-step fine-tuning. Considering the performance improvements shown in the preceding results, further tailoring of the pretraining pipeline to weather-specific tasks and additional refinement of the weather adaptations may close this gap and enable PDE foundation models to achieve operationally competitive weather forecasting performance.

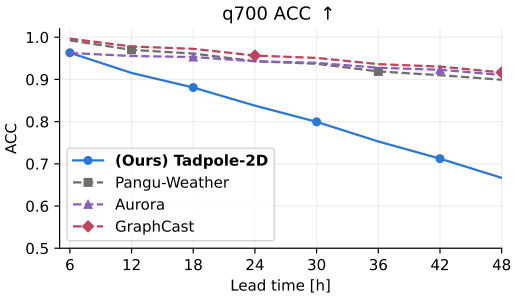


Figure 6: Comparison of Tadpole-2D's performance with Pangu-Weather, GraphCast, and Aurora, averaged over 730 forecasts in 2022.

4.2 PDE Dynamics

We compare the fine-tuning performance of Tadpole-2D with that of Poseidon-T and MORPH-S. The experimental setup is described in Appendix A.4.7. The models are rolled out autoregressively for 20 time steps. With one-step training, such extended rollouts can result in considerable autoregressive drift and smoothing. We quantify these effects using the normalized root-mean-square error NRMSE in pixel space and $\mathrm{NRMSE}_{\mathrm{Enstrophy}}$ in spectral space (Chen et al., 2024; Holzschuh et al., 2025). The definitions of these metrics are provided in Appendix A.4.8. Figure 7 compares the tested models at each time step and reports their average performance over the rollout period. Tadpole-2D attains the lowest rollout-mean error on both metrics with 8.2 million trainable parameters, compared with 20.8 million for Poseidon-T and 32.8 million for MORPH-S. On this task, too, Tadpole-2D outperforms the baselines while training the fewest parameters.

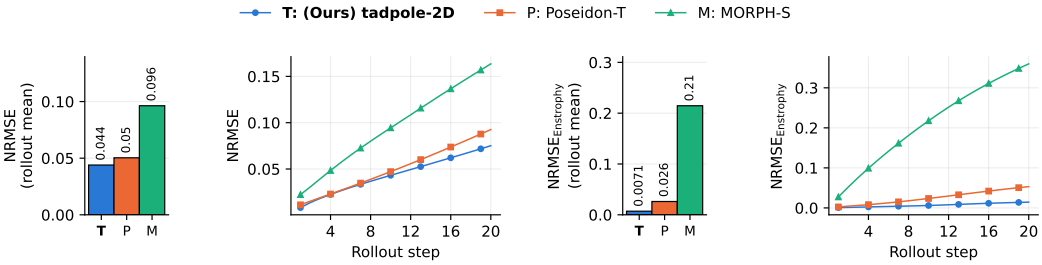


Figure 7: Comparison of Tadpole-2D with MORPH-S and Poseidon-T after fine-tuning on transonic cylinder flow dynamics. The Mach-number range of the test split lies outside that of the training set. Tadpole-2D performs best on both the spatial and spectral metrics.

5 CONCLUSION

Outlook Future work could incorporate additional variables at multiple pressure levels, surface-level variables, and static fields, enabling richer physical representations. The positional encoding could also include longitude, although we expect that static fields may already provide much of this location-specific information. Multi-step training and using multiple temporal snapshots as model input could stabilize long rollouts, allowing accurate long-range forecasts. Beyond weather forecasting, adapting to climate modelling is another challenging task to explore with PDE foundation models, where the model would need to reproduce long-term atmospheric statistics and responses to external forcings.

We have shown that a PDE foundation model pretrained with an online training pipeline and adapted to weather forecasting offers a route to easing the data and storage demands of AI-based weather model training. We have introduced a two-dimensional PDE foundation model, pretrained as a VAE on slices of 3D PDE data simulated on the fly with no storage requirements. The model adapts to short-range weather forecasting on the HEALPix grid and outperforms a cohort of similarly sized PDE foundation models on both weather prediction and PDE emulation.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on research methodology or experiments, and implement methods.

We have not used generative AI tools to propose or refine hypotheses, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, generate synthetic data sets, design research methodology or experiments, clean and reformat datasets, or interpret results. Assistance in the writing of proofs, assistance with translation, and support in qualitative and thematic data analysis are not applicable to this work.

Additionally, we used generative AI tools for editing software code, sourcing/searching for information, identifying relevant literature, summarizing or analysing existing literature, formatting references, proposing a title or keywords for the research paper, and paraphrasing parts of the research paper. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims, or artefacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we detail all network architectures, optimizers, training hyperparameters, train–test splits, batch sizes, number of training steps, hardware specifications, and other relevant details in the appendix. We also declare the datasets used in our experiments, along with the code to generate synthetic datasets. The source code for our experiments is included as supplementary material and is available in an anonymous GitHub repository: `https://anonymous.4open.science/r/tadpole2d-382D/`. It will be made publicly available upon acceptance.

REFERENCES

Peter Bauer, Alan Thorpe, and Gilbert Brunet. The quiet revolution of numerical weather prediction. *Nature*, 525(7567):47–55, 2015. doi: 10.1038/nature14956.

Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3D neural networks. *Nature*, 619:533–538, 2023. doi: 10.1038/s41586-023-06185-3.

Cristian Bodnar, Wessel P. Bruinsma, Ana Lucic, Megan Stanley, Anna Allen, Johannes Brandstetter, Patrick Garvan, Maik Riechert, Jonathan A. Weyn, Haiyu Dong, et al. A foundation model for the earth system. *Nature*, 641(8065):1180–1187, 2025. doi: 10.1038/s41586-025-09005-y.

Oscar Carlsson, Jan E. Gerken, Hampus Linander, Heiner Spieß, Fredrik Ohlsson, Christoffer Petersson, and Daniel Persson. HEAL-SWIN: A vision transformer on the sphere. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6067–6077, 2024.

Yifan Chen, Mark Goldstein, Mengjian Hua, Michael Samuel Albergo, Nicholas Matthew Boffi, and Eric Vanden-Eijnden. Probabilistic forecasting with stochastic interpolants and Föllmer processes. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 6728–6756, 2024. URL https://proceedings.mlr.press/v235/chen24n.html.

Krzysztof M. Górski, Eric Hivon, Anthony J. Banday, Benjamin D. Wandelt, Frode K. Hansen, Martin Reinecke, and Matthias Bartelmann. HEALPix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere. *The Astrophysical Journal*, 622(2): 759–771, 2005. doi: 10.1086/427976.

Insu Han, Rajesh Jayaram, Amin Karbasi, Vahab Mirrokni, David P. Woodruff, and Amir Zandieh. HyperAttention: Long-context attention in near-linear time. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=b0OhN0ii36.

Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. DPOT: Auto-regressive denoising operator transformer for large-scale PDE pre-training. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 17616–17635, 2024. URL https://proceedings.mlr.press/v235/hao24d.html.

Maximilian Herde, Bogdan Raonić, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel de Bézenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for PDEs. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-2311.

Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, Adrian Simmons, Cornel Soci, et al. The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146 (730):1999–2049, 2020. doi: 10.1002/qj.3803.

Benjamin Holzschuh, Qiang Liu, Georg Kohl, and Nils Thuerey. PDE-Transformer: Efficient and versatile transformers for physics simulations. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 23562–23602, 2025. URL https://proceedings.mlr.press/v267/holzschuh25a.html.

Benjamin Holzschuh, Georg Kohl, Florian Redinger, and Nils Thuerey. P3D: Scalable neural surrogates for high-resolution 3D physics simulations with global context. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.10186.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Matthias Karlbauer, Nathaniel Cresswell-Clay, Dale R. Durran, Raul A. Moreno, Thorsten Kurth, Boris Bonev, Noah Brenowitz, and Martin V. Butz. Advancing parsimonious deep learning weather prediction using the HEALPix mesh. *Journal of Advances in Modeling Earth Systems*, 16(8):e2023MS004021, 2024. doi: 10.1029/2023MS004021.

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014. URL https://arxiv.org/abs/1312.6114.

Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, volume 29, 2016. URL https://arxiv.org/abs/1606.04934.

Dmitrii Kochkov, Janni Yuval, Ian Langmore, Peter Norgaard, Jamie Smith, Grigory Mooers, Milan Klöwer, James Lottes, Stephan Rasp, Peter Düben, et al. Neural general circulation models for weather and climate. *Nature*, 632:1060–1066, 2024. doi: 10.1038/s41586-024-07744-y.

Georg Kohl, Li-Wei Chen, and Nils Thuerey. Benchmarking autoregressive conditional diffusion models for turbulent flow simulation. *Neural Networks*, 199:108641, 2026. doi: 10.1016/j.neunet.2026.108641.

Remi Lam, Álvaro Sánchez-González, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023. doi: 10.1126/science.adi2336.

Simon Lang, Mihai Alexe, Matthew Chantry, Jesper Dramsch, Florian Pinault, Baudouin Raoult, Mariana C. A. Clare, Christian Lessig, Michael Maier-Gerber, Linus Magnusson, et al. AIFS – ECMWF's data-driven forecasting system, 2024. URL https://arxiv.org/abs/2406.01465.

Hampus Linander, Christoffer Petersson, Daniel Persson, and Jan E. Gerken. PEAR: Equal area weather forecasting on the sphere. In *NeurIPS 2025 Workshop on AI for Science*, 2025. URL https://arxiv.org/abs/2505.17720.

Qiang Liu, Felix Koehler, and Nils Thuerey. TorchFSM: Fourier spectral method with PyTorch. Zenodo software release, May 2025. URL https://doi.org/10.5281/zenodo.15350210.

Qiang Liu, Felix Koehler, Benjamin Holzschuh, and Nils Thuerey. Tadpole: Autoencoders as foundation models for 3D PDEs with online learning, 2026. URL https://arxiv.org/abs/2605.15284.

Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer V2: Scaling up capacity and resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. URL https://arxiv.org/abs/2111.09883.

Michael McCabe, Payel Mukhopadhyay, Tanya Marwah, Bruno Régaldo-Saint Blancard, François Rozet, Cristiana Diaconu, Lucas Meyer, Kaze W. K. Wong, Hadi Sotoudeh, Alberto Bietti, et al. Walrus: A cross-domain foundation model for continuum dynamics. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026. URL https://arxiv.org/abs/2511.15684.

Lucas Thibaut Meyer, Marc Schouler, Robert Alexander Caulk, Alejandro Ribes, and Bruno Raffin. Training deep surrogate models with large scale online learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 24614–24630, 2023. URL https://proceedings.mlr.press/v202/meyer23b.html.

Tung Nguyen, Johannes Brandstetter, Ashish Kapoor, Jayesh K. Gupta, and Aditya Grover. ClimaX: A foundation model for weather and climate. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 25904–25938, 2023. URL https://proceedings.mlr.press/v202/nguyen23a.html.

Nathanaël Perraudin, Michaël Defferrard, Tomasz Kacprzak, and Raphael Sgier. DeepSphere: Efficient spherical convolutional neural network with HEALPix sampling for cosmological applications. *Astronomy and Computing*, 27:130–146, 2019. doi: 10.1016/j.ascom.2019.03.004.

Ilan Price, Álvaro Sánchez-González, Ferran Alet, Tom R. Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, Remi Lam, and Matthew Willson. Probabilistic weather forecasting with machine learning. *Nature*, 637(8044):84–90, 2025. doi: 10.1038/s41586-024-08252-9.

Vivek Ramavajjala. HEAL-ViT: Vision transformers on a spherical mesh for medium-range weather forecasting, 2024. URL https://arxiv.org/abs/2403.17016.

Stephan Rasp, Stephan Hoyer, Alexander Merose, Ian Langmore, Peter Battaglia, Tyler Russell, Álvaro Sánchez-González, Vivian Yang, Rob Carver, Shreya Agrawal, Matthew Chantry, Zied Ben Bouallègue, Peter Dueben, Carla Bromberg, Jared Sisk, Luke Barrington, Aaron Bell, and Fei Sha. WeatherBench 2: A benchmark for the next generation of data-driven global weather models. *Journal of Advances in Modeling Earth Systems*, 16(6):e2023MS004019, 2024. doi: 10.1029/2023MS004019.

Mahindra Singh Rautela, Alexander Most, Siddharth Mansingh, Bradley C. Love, Alexander Scheinker, Diane Oyen, Nathan Debardeleben, Earl Lawrence, and Ayan Biswas. MORPH: PDE foundation models with arbitrary data modality, 2025. URL https://arxiv.org/abs/2509.21670.

Irina Sandu. Destination Earth's digital twins and Digital Twin Engine – state of play. *ECMWF Newsletter*, 180:14–23, July 2024. doi: 10.21957/is1fc736jx.

Johannes Schmude, Sujit Roy, Will Trojak, Johannes Jakubik, Daniel Salles Civitarese, Shraddha Singh, Julian Kuehnert, Kumar Ankur, Aman Gupta, Christopher E. Phillips, et al. Prithvi WxC: Foundation model for weather and climate, 2024. URL https://arxiv.org/abs/2409.13598.

Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. URL https://arxiv.org/abs/1609.05158.

Zhanhong Ye, Zining Liu, Bingyang Wu, Hongjie Jiang, Leheng Chen, Minyan Zhang, Xiang Huang, Qinghe Meng, Jingyuan Zou, Hongsheng Liu, and Bin Dong. PDEformer-2: A versatile foundation model for two-dimensional partial differential equations, 2025. URL https://arxiv.org/abs/2507.15409.

A Appendix

A.1 Notation

A scalar quantity and its vector-valued or multichannel counterpart share a symbol, and the vector form is set in bold. Thus u is a scalar field and $\boldsymbol{u}$ a multivariate state, and v is a velocity component and $\boldsymbol{v}$ the velocity vector. The letter x is reserved for position, with $\boldsymbol{x}$ the position vector. Channel indices appear as parenthesized superscripts, so $u^{(c)}$ is channel c of the state $\boldsymbol{u}$. Physical time appears as a subscript. Hats mark model predictions, and a tilde on a latent marks the output of the latent dynamics network. Tensor shapes are written in brackets, for example $[B, C, H, W]$, and folding a dimension into the batch dimension is written as a product, for example $[B \cdot C, 1, H, W]$. Atmospheric variables are named in typewriter font, for example `z500` for geopotential at $500\,\mathrm{hPa}$. Table 1 collects the symbols used in the main text and in this appendix. Symbols that occur only inside a single derivation, such as the interpolation weights of Appendix A.4.5, are defined where they appear.

A.2 Network Architecture

The encoder combines a convolutional feature extractor with a hierarchical window-attention network. The convolutional stem has two spatial downsampling stages, followed by two further downsampling stages in the transformer. The decoder mirrors this sequence with learned upsampling. Attention is computed in non-overlapping 4×4 windows that are not shifted between blocks, so information crosses window borders through the downsampling convolutions.

We use the `B` configuration of P3D (Holzschuh et al., 2026). Its convolutional stem has widths 64, 128, and 128, and its transformer widens from 128 to 512 channels across the encoder. The transformer has five stages, each with two blocks and four attention heads: two encoder stages, one bottleneck stage, and two decoder stages. Table 2 lists every stage with the tensor it produces for a 64×64 pretraining crop and its parameter count. The autoencoder holds 19.75 million parameters, 10.42 million in the encoder and 9.33 million in the decoder.

Table 1: Notation used throughout the paper.

Symbol	Meaning
Fields and dimensions	
u	Scalar input field
$\hat{u}$	Reconstruction or prediction of u
$\boldsymbol{u}_t \in \mathbb{R}^{C\times H\times W}$	Multivariate state at time t
$u_t^{(c)}$	Channel c of $\boldsymbol{u}_t$, a scalar field
$\hat{\boldsymbol{u}}_{t+\Delta t}$	Predicted state one step ahead
$\boldsymbol{x}$	Position
$\boldsymbol{v} = (v_x, v_y, v_z)$	Velocity
B, b	Minibatch size and sample index
C, c	Number of physical channels and channel index
H, W	Height and width of a field or crop
t, Δt	Physical time and model time step ($\Delta t = 6\,\mathrm{h}$ for weather)
Model	
$\mathcal{E}$, $\mathcal{D}$	Pretrained encoder and decoder, applied per channel
$q_{\mathcal{E}}(\boldsymbol{z} \mid u)$	Latent distribution predicted by the encoder
$p_{\mathcal{D}}(u \mid \boldsymbol{z})$	Decoder likelihood
$p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$	Standard normal prior
$\boldsymbol{z}$	Latent of a single scalar field
$\boldsymbol{z}_t$	Channel-concatenated latent of $\boldsymbol{u}_t$
$\tilde{\boldsymbol{z}}_{t+\Delta t}$	Latent predicted by the dynamics network
G_θ	Latent dynamics subnetwork with parameters θ
α	Learned scale of the latent residual path
d	Number of latent scalars per field
Objectives and optimization	
$\mathcal{L}_{\mathrm{VAE}}$, $\mathcal{L}_{\mathrm{dyn}}$, $\mathcal{L}_{\mathrm{weather}}$	Pretraining, PDE dynamics, and weather objectives
λ_{KL}	Weight of the KL-divergence term
s	Learned scalar log variance of the decoder likelihood
κ	Free-bits threshold per latent coordinate
σ_c	Standard deviation of the six-hour difference of channel c
w_c	Loss weight of channel c
η_k, k	Learning rate at update k
η_0, η_f, T_d	Initial and final learning rates, decay length
Grids and evaluation	
N_{side}	Cells along each edge of a HEALPix base face
f	HEALPix base face index, $f \in \{1, \ldots, 12\}$
N_l	Cells along each edge of the latent grid of a face
ϕ, λ	Latitude and longitude
ω_i	Area weight of latitude row i
$\langle\cdot\rangle$	Area-weighted spatial mean
n, N_{init}	Forecast initialization index and number of initializations
t_n	Initialization time of forecast n
τ	Forecast lead time
$\bar{u}^{(c)}$	Climatology of channel c
$a_{n,\tau}^{(c)}$, $\hat{a}_{n,\tau}^{(c)}$	Observed and forecast anomalies

The encoder maps a scalar input to $\boldsymbol{z} \in \mathbb{R}^{512\times H/16\times W/16}$; hence, a 64×64 pretraining crop produces a $512 \times 4 \times 4$ latent. This latent has 16 times fewer grid points along each spatial axis than the input, but it holds twice as many scalars, $8{,}192$ compared with $4{,}096$. Tadpole-2D is therefore a spatial bottleneck and not a compressive one. A final 1×1 convolution predicts the latent means and log variances. Pretraining uses circular convolutional padding and does not use encoder-decoder skip connections, forcing the latent representation to learn the reconstruction. The convolutional and windowed design allows larger grids to be processed. When a field is partitioned into crops, we encode its scalar crop-channel pairs independently and restore their spatial arrangement before applying a downstream latent model.

Table 2: Stages of the Tadpole-2D autoencoder. Output shapes are given as $C \times H \times W$ for a single-channel 64×64 pretraining crop; a batch of B fields with C channels is processed as $B \cdot C$ such crops.

Stage	Output shape	Parameters
Encoder		
Input field	$1 \times 64 \times 64$	–
Stem embedding, 3×3 convolution	$64 \times 64 \times 64$	640
Downsampling convolution, stride 2	$128 \times 32 \times 32$	73,856
Residual convolution blocks ($\times 2$)	$128 \times 32 \times 32$	591,360
Downsampling convolution, stride 2	$128 \times 16 \times 16$	147,584
Transformer stage, 16 windows	$128 \times 16 \times 16$	369,928
Downsampling convolution, stride 2	$256 \times 8 \times 8$	294,912
Transformer stage, 4 windows	$256 \times 8 \times 8$	1,453,576
Downsampling convolution, stride 2	$512 \times 4 \times 4$	1,179,648
Bottleneck transformer stage, 1 window	$512 \times 4 \times 4$	5,783,560
Latent head, 1×1 convolution (means, log variances)	$1024 \times 4 \times 4$	525,312
Encoder total		10,420,376
Decoder		
Latent $\boldsymbol{z}$ sampled from $q_{\mathcal{E}}(\boldsymbol{z} \mid u)$	$512 \times 4 \times 4$	–
Upsampling block, convolution and pixel shuffle	$256 \times 8 \times 8$	5,308,416
Transformer stage, 4 windows	$256 \times 8 \times 8$	1,453,576
Upsampling block, convolution and pixel shuffle	$128 \times 16 \times 16$	1,327,104
Transformer stage, 16 windows	$128 \times 16 \times 16$	369,928
Transformer output layer, normalization and convolution	$128 \times 16 \times 16$	180,608
Upsampling block, 1×1 convolution and pixel shuffle	$128 \times 32 \times 32$	66,048
Residual convolution blocks ($\times 2$)	$128 \times 32 \times 32$	591,360
Upsampling block, 1×1 convolution and pixel shuffle	$64 \times 64 \times 64$	33,024
Output projection, 3×3 convolution	$1 \times 64 \times 64$	577
Decoder total		9,330,641
Autoencoder total		19,751,017

Convolutional Blocks Spatial downsampling is always a 3×3 convolution with stride two. A residual convolution block applies group normalization with 32 groups, a GELU nonlinearity, and a 3×3 convolution twice, then adds its input; two such blocks sit at the intermediate resolution of the stem and, mirrored, of the convolutional decoder. Upsampling doubles the width with a convolution and then redistributes the extra channels over space with a pixel shuffle (Shi et al., 2016). Inside the transformer, the upsampling block wraps this pair between two 3×3 convolutions; in the convolutional decoder a 1×1 convolution suffices.

Transformer Blocks Each transformer block is pre-normalized: layer normalization precedes window attention and also precedes a two-layer MLP with GELU and an expansion ratio of four, with a residual connection around each. Queries, keys, and values come from a single linear projection without bias, and the attention output enters the residual stream without a further projection. Attention logits are inner products of ℓ_2-normalized queries and keys scaled by a learned per-head factor, and the relative position bias within a window is generated by a small MLP of the relative coordinates, both as in Swin Transformer V2 (Liu et al., 2022). Because the window is 4×4, a 64×64 crop is partitioned into 16, 4, and 1 windows at the three transformer resolutions, so the bottleneck stage attends over the whole crop.

Latent Head The 1×1 convolution of the latent head doubles the bottleneck width to 1024 channels, which are read as the means and log variances of a diagonal Gaussian over the $d = 8{,}192$ latent coordinates of a 64×64 crop. Log variances are clamped to $[-30, 20]$. During pretraining, the decoder receives a reparameterized sample from this distribution, and the same log variances enter the free-bits term of Appendix A.4.1.

Padding Every convolution in the backbone takes its padding rule from a single setting, so the same weights can be evaluated under different domain topologies. Pretraining uses circular padding

throughout, matching the periodic domains of the pretraining simulations. Section 3.2 replaces it with a HEALPix-aware padding that gathers each face's neighbours, leaving the parameter count unchanged.

A.3 Latent Dynamics Subnetwork

The dynamics subnetwork G_θ inserted at the bottleneck (Section 3.2.1) is a small transformer over the cells of the latent grid. The same architecture, with six layers of width 176 and eight attention heads, is used for the weather and the transonic cylinder-flow tasks; only its input width depends on the number of physical channels.

Input The encoder processes each channel $u_t^{(c)}$ separately and returns a sample from $q_\mathcal{E}(\boldsymbol{z} \mid u_t^{(c)})$ with 512 latent channels. These per-channel latents are concatenated along the latent channel axis into $\boldsymbol{z}_t \in \mathbb{R}^{512C \times N_l \times N_l}$, so the physical variables interact only inside G_θ. For weather, $C = 5$ and every HEALPix base face is a separate sample with a 4×4 latent grid; for the transonic cylinder flow, $C = 4$ and the whole 128×64 field gives an 8×4 latent grid. Each latent grid cell becomes one token with $512C$ features. Attention therefore acts within a face and never across faces; information crosses face boundaries only through the HEALPix-aware convolutions of the encoder and decoder.

Tokens and Positional Information A linear layer projects each token to width 176. A fixed two-dimensional sine–cosine embedding of the token's row and column indices on the latent grid is added. For weather, the HEALPix coordinate embedding of Section 3.2 is added at the same point. A single learned token, initialized with standard deviation 10^{-6}, is appended to the sequence. It takes part in attention in every layer and is discarded before the output projection.

Transformer Layers Each of the six layers is pre-normalized with RMS normalization and has residual connections around its attention and feed-forward sublayers,

$$\begin{aligned} \boldsymbol{h}' &= \boldsymbol{h} + \mathrm{Attn}\big(\mathrm{RMSNorm}(\boldsymbol{h})\big), \\ \boldsymbol{h}'' &= \boldsymbol{h}' + \boldsymbol{W}_{\mathrm{down}}\Big(\mathrm{GELU}\big(\boldsymbol{W}_{\mathrm{gate}}\,\mathrm{RMSNorm}(\boldsymbol{h}')\big) \odot \boldsymbol{W}_{\mathrm{up}}\,\mathrm{RMSNorm}(\boldsymbol{h}')\Big). \end{aligned} \tag{4}$$

The gated feed-forward part has an inner width of $4 \cdot 176 = 704$ and no biases. Attention is bidirectional with eight heads of width 22; queries, keys, and values come from one linear projection with bias, and the heads are merged by an output projection with bias. We use the HyperAttention implementation (Han et al., 2024), which computes exact softmax attention for sequences shorter than 4,096 tokens. With 17 tokens per weather face and 33 per transonic field, all attention in our experiments is exact.

Output and Initialization A final linear layer maps each token back to $512C$ features, and the tokens are rearranged into the $512C \times N_l \times N_l$ latent grid. The latent update is

$$\tilde{\boldsymbol{z}}_{t+\Delta t} = \alpha\, \boldsymbol{z}_t + G_\theta(\boldsymbol{z}_t), \tag{5}$$

with a learned scalar α initialized to one. The weights and bias of the output layer are initialized to zero, so $G_\theta(\boldsymbol{z}_t) = \mathbf{0}$ and $\tilde{\boldsymbol{z}}_{t+\Delta t} = \boldsymbol{z}_t$ at the start of fine-tuning. The decoder then maps each channel's 512-channel block of $\tilde{\boldsymbol{z}}_{t+\Delta t}$ back to physical space. Table 3 lists the layers for a weather face.

A.4 Training and Algorithmic Details

A.4.1 Pretraining Details

Equation 1 is optimized in the following expanded form. For a minibatch of B scalar fields of size $H \times W$, let d denote the number of latent scalars per field, let $\hat{u}_b$ be the reconstruction of field u_b, and let s be a learned scalar log variance. We minimize

$$\mathcal{L}_{\mathrm{VAE}} = \exp(-s)\frac{1}{B}\sum_{b=1}^{B}\frac{\|\hat{u}_b - u_b\|_2^2}{HW} + s + \frac{\lambda_{\mathrm{KL}}}{Bd}\sum_{b=1}^{B}\sum_{j=1}^{d}\max\left(D_{\mathrm{KL}}\left(q_\mathcal{E}(z_{bj} \mid u_b)\,\|\,\mathcal{N}(0,1)\right), \kappa\right), \tag{6}$$

Table 3: Latent dynamics subnetwork G_θ for one weather face ($C = 5$, 4×4 latent grid). Output shapes are given as tokens $\times$ features. For the transonic cylinder flow ($C = 4$, 8×4 latent grid), the input and output projections contain 360,624 and 362,496 parameters, for a total of 3,703,328.

Stage	Output shape	Parameters
Latent $\boldsymbol{z}_t$, flattened to tokens	16×2560	–
Input projection, linear	16×176	450,736
Sine–cosine positional embedding, fixed	16×176	0
Appended learned token	17×176	176
Transformer layers ($\times 6$)	17×176	2,980,032
RMS normalization ($\times 2$ per layer)		2×176
Multi-head attention, 8 heads		124,608
Gated feed-forward, width 704		371,712
Learned token removed	16×176	–
Output projection, linear, zero-initialized	16×2560	453,120
Subnetwork total		3,884,064
HEALPix coordinate embedding, 1×1 convolution	16×176	528
Latent residual scale α	–	1

with $\lambda_{\mathrm{KL}} = 10^{-6}$ and s initialized at 0.

The first two terms are the reconstruction likelihood of Equation 1 written out. For a Gaussian decoder with a single learned scalar variance, $\mathbb{E}_{q_\mathcal{E}(\boldsymbol{z}|u)}[-\log p_\mathcal{D}(u \mid \boldsymbol{z})]$ equals $\exp(-s)\,\mathrm{MSE} + s$ up to an additive constant and an overall factor. Here $\exp(-s)$ acts as a learned precision that rescales the mean squared error, while the bare s penalizes inflating that variance to make the first term cheap.

The third term applies free bits (Kingma et al., 2016). The threshold $\kappa = 0.125$ is applied to each latent coordinate separately before averaging, so a coordinate whose divergence from the prior has already fallen below κ contributes no gradient to the regularizer.

For the KL divergence, we use a weighting of $\lambda_{\mathrm{KL}} = 10^{-6}$. The model is optimized with AdamW at a learning rate of 10^{-4}, a global batch size of 750, and 1,000 warmup steps, using mixed bfloat16 precision. We keep an exponential moving average of the weights with decay 0.999, and all downstream experiments in this paper start from that averaged checkpoint.

A.4.2 Normalized Root-Mean-Square Error

The pointwise scores are normalized root-mean-square errors (NRMSEs) in physical units. Every evaluation frame is min–max normalized before it enters the network, and both the prediction and the target are mapped back to physical units before the error is formed, so the score does not depend on each dataset's normalization constants.

Consider a dataset of N evaluation frames, each holding C channels on an $H \times W$ grid. Write $u_n^{(c)}$ for channel c of frame n and $\hat{u}_n^{(c)}$ for its prediction. We form a relative error per channel on each frame, average it over the channels of that frame, and then average over frames:

$$\mathrm{NRMSE} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{C}\sum_{c=1}^{C}\frac{\sqrt{\frac{1}{HW}\sum_{\boldsymbol{x}}\left(\hat{u}_n^{(c)}(\boldsymbol{x}) - u_n^{(c)}(\boldsymbol{x})\right)^2}}{\sqrt{\frac{1}{HW}\sum_{\boldsymbol{x}}\left(u_n^{(c)}(\boldsymbol{x})\right)^2}}, \tag{7}$$

where both sums over $\boldsymbol{x}$ run over the HW grid points of the frame.

The normalization is applied per channel and before the channel average, so every channel carries the same weight regardless of its physical magnitude; flattening channels and spatial positions together before normalizing would instead let the channel with the largest magnitude dominate the score.

A.4.3 Weather Forecasting Experiments: Fine-Tuning versus Training from Scratch

The comparison in Section 4.1 consists of six runs: two initializations crossed with three training intervals. The fine-tuned runs load the pretrained Tadpole-2D encoder and decoder, freeze their weights, and train rank-32 low-rank updates together with the dynamics subnetwork. The from-scratch runs use the same architecture without low-rank adaptation, and train all encoder, decoder, and dynamics weights from random initialization. The two variants differ only in these settings.

Every training interval ends with the last six-hourly frame of 2018 and extends backward by a whole number of calendar years, as listed in Table 4. These runs use the same five variables, per-variable standardization, and time-difference based loss weighting of Section 3.2.

Table 4: Training intervals for the fine-tuning versus training-from-scratch comparison. Each run takes $6{,}900$ updates with 64 samples per update, so the number of passes over the training data decreases as the interval grows.

Data budget	Training years	Six-hourly frames	Passes over the data
1 year	2018	1,460	≈ 300
4 years	2015–2018	5,844	≈ 76
16 years	2003–2018	23,376	≈ 19

Table 5 lists the shared optimization settings. Each run trains on single-step forecast samples, so no autoregressive unrolling occurs during training. The learning rate decays exponentially from its initial value η_0 to its final value η_f over the first T_d updates and then stays constant, with no warmup:

$$\eta_k = \eta_0 \left(\frac{\eta_f}{\eta_0} \right)^{\min(k,\, T_d)/T_d}, \tag{8}$$

where k counts optimizer updates. We keep an exponential moving average of the trainable weights and evaluate the averaged weights after the final update. The validation loss is recorded every $1{,}000$ updates but is not used to select checkpoints. Each run uses a single NVIDIA GH200 GPU and random seed 0.

Table 5: Optimization settings shared by all runs in the fine-tuning versus training-from-scratch comparison.

Setting	Value
Optimizer	AdamW, weight decay 10^{-15}
Updates	6,900
Batch size	64
Initial learning rate η_0	10^{-3}
Final learning rate η_f	10^{-4}
Decay length T_d	3,000 updates
Warmup	none
Training rollout length	one six-hour step
Precision	mixed bfloat16
Weight averaging	exponential moving average, decay 0.999
Low-rank adaptation rank	32 (fine-tuned runs only)
Trainable parameters	6.16M (fine-tuned), 23.63M (from scratch)

A.4.4 Weather Forecasting Experiments: Comparison between PDE Foundation Models

The three PDE foundation models in the comparison of Section 4.1, Poseidon-T (Herde et al., 2024), MORPH-S (Rautela et al., 2025), and PDEformer-2-S (Ye et al., 2025), are fine-tuned with the default recipes in their fine-tuning documentation, that is, with the loss, optimizer, learning-rate schedule, and gradient clipping of the fine-tuning scripts and configurations in their public code

releases. Every model starts from its released pretrained weights; for PDEformer-2 we convert the weights to a PyTorch reimplementation. All four models are fine-tuned on the same one-step forecasting task on the `HPX64` grid, with the same data, batch size, and number of updates.

Shared Training Setup All runs train on 1979–2018 and validate on 2019. They use the variables, `HPX64` remapping, and per-variable standardization of Section 3.2. Table 6 lists the settings that are identical across the four runs. We keep these settings common, instead of taking each repository's own values, so that every model sees the same number of training samples. With 12 samples per update, the 40,000 updates cover 480,000 samples, or about eight passes over the 58,439 training samples. The validation loss is recorded every 2,000 updates but is not used to select checkpoints.

Table 6: Settings shared by all runs in the comparison with PDE foundation models. The loss, optimizer, learning-rate schedule, and gradient clipping differ between the models and are listed in Table 7.

Setting	Value
Updates	40,000
Batch size	12 (144 face images)
Training rollout length	one six-hour step
Precision	mixed bfloat16
Weight averaging	exponential moving average, decay 0.999
Hardware	one NVIDIA GH200 GPU
Random seed	0

Training Recipes Table 7 lists the recipe of each model. Tadpole-2D minimizes the tendency-weighted objective of Equation 3, and its learning rate follows Equation 8. Poseidon minimizes the relative L^1 error of each channel, that is, the mean absolute error of the channel over the batch divided by the mean absolute value of its target, averaged over the five channels. Poseidon's recipe groups channels that form one physical vector; we use one group per channel. Its optimizer applies no weight decay to biases and layer-norm parameters, and trains the embedding layers, the patch recovery, and the time-conditioned layer norms with a ten times larger learning rate than the rest of the network. The learning rate of every group decays to zero along a cosine over the whole run. MORPH minimizes the mean squared error with a constant learning rate that is halved when the validation loss has not improved for five consecutive validations. We step this schedule once per validation, as in MORPH's training loop. The validation loss improved at every validation, so the learning rate stayed at 10^{-4} throughout. PDEformer-2 minimizes a normalized root-mean-square error: for each face, the root-mean-square error over all five channels is divided by the root mean square of the target plus 0.05, and the result is averaged over the faces. Its learning rate rises linearly over the first 400 updates, or 1% of the run, and then decays along a cosine towards 10^{-6}.

Table 7: Fine-tuning recipes in the comparison with PDE foundation models. Tadpole-2D uses its own recipe; each baseline uses the default fine-tuning recipe of its public code release. Gradient clipping bounds the global norm of the gradient. [a]Embedding layers, patch recovery, and time-conditioned layer norms.

	Tadpole-2D	Poseidon-T	MORPH-S	PDEformer-2-S
Trainable parameters	8.4M	20.8M	32.8M	27.7M
Fine-tuning	LoRA, rank 64	full	full	full
Loss	weighted MSE	relative L^1	MSE	normalized RMSE
Optimizer	AdamW	AdamW	AdamW	Adam
Weight decay	10^{-15}	10^{-6}	0	0
Initial learning rate	10^{-3}	5×10^{-5}, 5×10^{-4} [a]	10^{-4}	10^{-4}
Schedule	exponential	cosine	on plateau	cosine
Final learning rate	10^{-4}	0	10^{-4}	10^{-6}
Warmup	none	none	none	400 updates
Gradient clipping	none	5	none	1

Evaluation Each model is evaluated in single precision on the 730 forecasts initialized at 00:00 and 12:00 UTC in 2022, using the metrics of Appendix A.5.

For every model, the twelve base faces are folded into the batch dimension and processed independently. Each baseline maps the five-channel face at time t to the five-channel face at $t + \Delta t$, and its output is used as the forecast.

Tadpole-2D Tadpole-2D uses the adaptation of Section 3.2.1 and the HEALPix interface of Section 3.2, with low-rank adaptation of rank 64 instead of 32. This raises the number of trainable parameters from 6.16 million to 8.4 million, while the 19.7 million weights of the frozen pretrained backbone stay unchanged.

Poseidon-T We fine-tune all weights of Poseidon-T. Its patch embedding and patch recovery are sized for the four channels of its pretraining data. As in Poseidon's own fine-tuning procedure, the weights of these layers are newly initialized for the five ERA5 variables, and all other weights retain their pretrained values. Poseidon was pretrained on 128×128 inputs; as in its fine-tuning scripts, the model resolution is reset to the downstream task's data, so a 64×64 face with patch size 4 gives a 16×16 token grid. Its lead-time input is 0.1 for every six-hour step.

MORPH-S We fine-tune all weights of MORPH-S and keep its dropout rate of 0.1. MORPH represents its input as a set of fields with up to three components each; we pass `z500`, `t850`, and `q700` as three scalar fields and (`u850`, `v850`) as one two-component vector field. With patch size 8, each face gives an 8×8 token grid. MORPH has no time input.

PDEformer-2-S PDEformer-2 encodes a PDE as a symbolic computational graph whose nodes may carry functions sampled on a 128×128 grid, and it decodes the solution with an implicit neural representation queried at space-time coordinates. We describe the forecast step with the graph

$$\partial_t u_i + f_i(u_1, \ldots, u_5) = 0, \qquad i = 1, \ldots, 5, \tag{9}$$

where each f_i is PDEformer-2's placeholder for an unspecified term and the current face state provides the initial conditions u_i. The faces are upsampled bilinearly to the 128×128 function grid, and the implicit neural representation is queried at the 64×64 cell centres at time 0.1, the same scaled lead time as for Poseidon. PDEformer-2's recipe evaluates the training loss at $8{,}192$ randomly sampled space-time points; we instead query all $4{,}096$ cell centres of the face, because the forecast is needed on the full face.

A.4.5 Remapping between HEALPix and Latitude-Longitude Grids

The appropriate remapping rule depends on whether the target grid refines or coarsens the source grid. For refinement, we use bilinear interpolation to estimate the field at each target-cell centre from nearby source values. For coarsening, we instead use first-order conservative remapping, which treats source values as cell averages and weights them by their overlap with each target cell. Bilinear interpolation would sample only a few source cells and can therefore discard resolved variability during coarsening. WeatherBench 2 likewise uses first-order conservative remapping to place forecasts and ground truth on a common 1.5° grid before computing its reported metrics (Rasp et al., 2024).

Our pipeline involves three grids. ERA5 is stored on a 0.25° latitude–longitude grid with 721×1440 cells, the model operates on the $N_{\text{side}} = 64$ HEALPix grid with $49{,}152$ cells, and evaluation uses the 1.5° latitude–longitude grid with 121×240 cells. The last is the resolution used for the WeatherBench 2 paper's headline comparisons; the current public scorecards are available online.[2] Both transformations in the main pipeline are therefore coarsenings. We map ERA5 to the model grid with a sampled first-order conservative map, using 32×32 area-uniform samples within each source cell. We map model forecasts directly from HEALPix to the evaluation grid with the corresponding conservative construction, using 32×32 samples within each target cell. The direct second map avoids introducing an intermediate latitude–longitude field.

[2] `https://sites.research.google/gr/weatherbench/deterministic-scores/`

Bilinear Regridding For a target centre (ϕ, λ) bracketed by source latitudes $\phi_0 \le \phi \le \phi_1$ and longitudes $\lambda_0 \le \lambda \le \lambda_1$, define

$$a = \frac{\phi - \phi_0}{\phi_1 - \phi_0}, \qquad b = \frac{\lambda - \lambda_0}{\lambda_1 - \lambda_0}. \tag{10}$$

The longitude interval is chosen periodically across the 0° seam. The interpolated value is

$$\begin{aligned} \widetilde{u}(\phi, \lambda) = (1-a)(1-b)u_{00} + (1-a)bu_{01} \\ + a(1-b)u_{10} + abu_{11}, \end{aligned} \tag{11}$$

where $u_{rs} = u(\phi_r, \lambda_s)$. Thus each target value is a convex combination of four neighbouring source values. For the reverse direction, we take the analogous four-cell HEALPix interpolation weights from the `healpy` package's `get_interp_weights`. Bilinear regridding is available for refinement paths, but neither transformation in the main evaluation path is a refinement.

Conservative Regridding Let S_i and T_j denote source and target cells on the sphere, with the source field represented by the cell averages u_i. First-order conservative regridding computes

$$\widetilde{u}_j = \sum_i r_{ji} u_i, \qquad r_{ji} = \frac{\operatorname{area}(S_i \cap T_j)}{\sum_k \operatorname{area}(S_k \cap T_j)}. \tag{12}$$

Each row of weights sums to one, so the map preserves constant fields while averaging all source cells that overlap a target cell. Exact spherical overlap polygons between latitude–longitude boxes and curved HEALPix cells are awkward to compute. We approximate their areas by uniform sampling in longitude and $\sin(\phi)$, which gives equal-area samples on the sphere. For latitude–longitude to HEALPix, each sample from a source cell contributes to the HEALPix cell containing its centre; for the reverse direction, samples within a target latitude–longitude cell select their containing HEALPix cells. Aggregating these assignments and normalizing each target row gives the sparse weights in Equation 12.

A.4.6 Transonic Cylinder Flow: Training Objective

The fine-tuning runs of Section 3.3 minimize a one-step prediction error with a uniform pixelwise mean squared error. Let $u^{(c)}_{t+\Delta t}$ be channel c of the state one frame ahead, $\hat{u}^{(c)}_{t+\Delta t}$ the prediction of the model, and $H \times W = 128 \times 64$ the field size. The objective is

$$\mathcal{L}_{\mathrm{dyn}} = \frac{1}{C} \sum_{c=1}^{C} \frac{1}{HW} \left\| \hat{u}^{(c)}_{t+\Delta t} - u^{(c)}_{t+\Delta t} \right\|_2^2, \tag{13}$$

with $C = 4$ for the channels (v_x, v_y, ρ, p) and $\Delta t = 0.002$ the frame interval of the dataset. The error is formed in the min–max normalized units in which the fields enter the network. Every channel and every grid point carries the same weight: unlike the weather objective of Equation 3, no per-channel weighting is applied.

A.4.7 Transonic Cylinder Flow: Comparison with PDE Foundation Models

The comparison in Section 3.3 fine-tunes Tadpole-2D and two pretrained PDE foundation models, Poseidon-T (Herde et al., 2024) and MORPH-S (Rautela et al., 2025), for one-step prediction of the transonic cylinder flow problem. The dataset is described in Appendix A.6.2. Every model starts from its released pretrained weights. The three runs share one training pipeline, data split, loss, and set of optimization settings, and differ only in the network.

Shared Training Setup All runs train on the 11 trajectories with Mach numbers 0.53–0.63 and validate on frames 500–749 of the three trajectories with Mach numbers 0.50–0.52 (Section 3.3). Every pair of consecutive frames of a training trajectory is one sample, which gives $11{,}000$ training samples. Each channel is min–max normalized to $[-1, 1]$ with constants stored with the dataset, and every model minimizes the uniform one-step MSE of Equation 13 on the full 128×64 field. The loss includes the cells inside the cylinder. Table 8 lists the optimization settings, which are identical across the three runs. The learning rate follows Equation 8, with the decay spanning the whole run. With 120 samples per update, the $20{,}000$ updates cover 2.4 million samples, or about 218 passes over the training set. The validation loss is recorded every $1{,}840$ updates, but is not used to select checkpoints.

Table 8: Optimization settings shared by all runs in the comparison with PDE foundation models on the transonic cylinder flow.

Setting	Value
Optimizer	AdamW, weight decay 10^{-15}
Updates	20,000
Batch size	120
Initial learning rate η_0	10^{-3}
Final learning rate η_f	10^{-5}
Decay length T_d	20,000 updates
Warmup	none
Gradient clipping	global norm 1
Training rollout length	one frame ($\Delta t = 0.002$)
Precision	mixed bfloat16
Weight averaging	exponential moving average, decay 0.999
Hardware	one NVIDIA GH200 GPU
Random seed	0

Evaluation Each model is evaluated with the averaged weights after the final update, in single precision. The test set holds the five trajectories with Mach numbers 0.64–0.68, which lie above the training range. Every frame from 500 to 729 of these trajectories starts one 20-step autoregressive rollout, so the last predicted frame is at most frame 749. This gives 1,150 rollouts. We report both metrics of Appendix A.4.8 at each rollout step, with $N = 1{,}150$ rollouts and $C = 4$ channels.

Models Table 9 summarizes the three models. Each network maps the four-channel state (v_x, v_y, ρ, p) at time t to the state at $t + \Delta t$.

Table 9: Models in the comparison with PDE foundation models on the transonic cylinder flow. The token grid is that of the 128×64 field, or, for Poseidon, of the padded 128×128 field.

Model	Trainable parameters	Fine-tuning	Token grid	Lead-time input
Tadpole-2D	8.2M	LoRA, rank 64	8×4	none
Poseidon-T	20.8M	full	32×32	0.002
MORPH-S	32.8M	full	16×8	none

Tadpole-2D Tadpole-2D uses the adaptation of Section 3.2.1, with low-rank adaptation of rank 64 instead of 32. The encoder processes the whole 128×64 field of each channel in one pass, which gives an 8×4 latent grid, and the latent subnetwork receives $4 \times 512 = 2{,}048$ input channels. Pretraining used circular padding, whereas here all convolutions use zero padding, because the domain is not periodic. The run trains 8.2 million parameters, while the 19.7 million weights of the pretrained backbone stay frozen.

Poseidon-T We fine-tune all weights of Poseidon-T. Poseidon was pretrained on four channels (ρ, v_x, v_y, p) at 128×128, so we reorder the channels to this order and keep the pretrained patch embedding and patch recovery. Poseidon accepts only square inputs, so we extend the field to 128×128 by replicating its edge values along the short axis, and crop the output back to 128×64 before the loss and the metrics. With patch size 4, this gives the 32×32 token grid of pretraining. The lead-time input is the frame interval $\Delta t = 0.002$ for every step.

MORPH-S We fine-tune all weights of MORPH-S and keep its dropout rate of 0.1. We pass (v_x, v_y) as one two-component vector field, and density and pressure as two scalar fields, each of depth one. With patch size 8, the field gives a 16×8 token grid. MORPH has no time input.

A.4.8 Evaluation Metrics for the PDE Dynamics Task

The rollouts of Section 3.3 are scored with two metrics: a pointwise NRMSE and a spectral error that records how faithfully the distribution of enstrophy over spatial scales is reproduced. Both are reported at every rollout step. Predictions and references are mapped back to physical units, so the scores are independent of the normalization constants. Both metrics also exclude the cells inside the cylinder, using an obstacle mask that is one for a fluid cell and zero for a cell inside the cylinder, although they apply it at different points of the computation.

Pointwise Error The pointwise score is the NRMSE of Equation 7, with N the number of rollouts of the evaluation set. A relative error is formed per channel, averaged over the four channels, and then averaged over rollouts. An obstacle mask is applied to the predictions and targets to nullify the obstacle-cell's contributions.

Spectral Error The spectral score compares the enstrophy spectrum of the prediction with that of the reference. For a single frame, we form the out-of-plane vorticity

$$\omega = \frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y} \tag{14}$$

from the two velocity channels by finite differences on the grid, and then apply the obstacle mask. This order prevents a spurious shear layer at the cylinder boundary and unphysical energy at high wavenumbers. The masked vorticity is multiplied by a separable Hann window, which is itself divided by its root mean square so that the window preserves the total energy of the field, and transformed with a two-dimensional FFT. Writing $\hat{\omega}(\boldsymbol{k})$ for the transform of one frame, normalized by the number of grid points, the shell spectrum is

$$S(k) = \sum_{\boldsymbol{k}\,:\,|\boldsymbol{k}|\in[k-1/2,\;k+1/2)} \tfrac{1}{2}\,|\hat{\omega}(\boldsymbol{k})|^2\,, \tag{15}$$

that is, the enstrophy power is binned into annular shells of unit width in $|\boldsymbol{k}|$. Shells above two thirds of the largest wavenumber resolved along the shorter axis of the 128×64 grid are discarded.

The error is then taken between mean spectra rather than averaged over rollouts. For one rollout step, let $\bar{S}_{\mathrm{pred}}(k)$ and $\bar{S}_{\mathrm{ref}}(k)$ be the spectra of Equation 15 averaged over all rollouts of the evaluation set. The spectral error is

$$\mathrm{NRMSE}_{\mathrm{Enstrophy}} = \sqrt{\frac{\sum_k \big(\bar{S}_{\mathrm{pred}}(k) - \bar{S}_{\mathrm{ref}}(k)\big)^2}{\sum_k \bar{S}_{\mathrm{ref}}(k)^2}}, \tag{16}$$

where both sums run over the retained shells. Averaging the spectra before forming the error separates a systematic spectral bias, such as a model that damps the small scales, from the per-sample variation that chaotic dynamics produce at long rollout horizons; averaging per-rollout spectral errors would mix the two. Because Equation 15 is a power rather than an amplitude spectrum, $\mathrm{NRMSE}_{\mathrm{Enstrophy}}$ scales quadratically in the amplitude of the field even with the square root of Equation 16. Its values should therefore be read relative to each other and not against the pointwise NRMSE.

A.5 Evaluation Metrics for the Weather Forecasting Experiments

The metrics follow the deterministic headline scores of WeatherBench 2 (Rasp et al., 2024). All quantities are evaluated on the 1.5° (121×240) equiangular grid with poles. For grid compatibility, we remap conservatively from `HPX64` (Appendix A.4.5) to 1.5° (121×240) and calculate errors against the WeatherBench 2 ERA5 fields on the same grid. Forecasts are computed in single precision using the averaged weights of Appendix A.4.3.

RMSE Let ϕ_i^- and ϕ_i^+ be the latitudes of the northern and southern edges of the cells in row i, and let N_λ be the number of longitudes. Every cell in row i carries the spherical area weight

$$\omega_i = \frac{|\sin\phi_i^+ - \sin\phi_i^-|}{N_\lambda \sum_{i'} |\sin\phi_{i'}^+ - \sin\phi_{i'}^-|}, \qquad \sum_{i,j} \omega_i = 1, \tag{17}$$

and we write $\langle a \rangle = \sum_{i,j} \omega_i \, a_{ij}$ for the weighted spatial mean of a field a. Let $\hat{u}^{(c)}_{n,\tau}$ be the forecast of channel c at lead time τ from initialization n, let $u^{(c)}_{n,\tau}$ be the verifying ERA5 field, and let $N_{\text{init}} = 730$ be the number of initializations. The root-mean-square error is

$$\mathrm{RMSE}_c(\tau) = \sqrt{\frac{1}{N_{\text{init}}} \sum_{n=1}^{N_{\text{init}}} \left\langle \left(\hat{u}^{(c)}_{n,\tau} - u^{(c)}_{n,\tau} \right)^2 \right\rangle}. \tag{18}$$

Following the WeatherBench 2 default, the square root is taken after averaging the squared error over all initializations, not per forecast.

ACC The anomaly correlation coefficient (ACC) compares the forecast and observed departures from a climatology $\bar{u}$,

$$\mathrm{ACC}_c(\tau) = \frac{1}{N_{\text{init}}} \sum_{n=1}^{N_{\text{init}}} \frac{\left\langle \hat{a}^{(c)}_{n,\tau} \, a^{(c)}_{n,\tau} \right\rangle}{\sqrt{\left\langle \left(\hat{a}^{(c)}_{n,\tau} \right)^2 \right\rangle \left\langle \left(a^{(c)}_{n,\tau} \right)^2 \right\rangle}}, \qquad \begin{aligned} \hat{a}^{(c)}_{n,\tau} &= \hat{u}^{(c)}_{n,\tau} - \bar{u}^{(c)}(t_n + \tau), \\ a^{(c)}_{n,\tau} &= u^{(c)}_{n,\tau} - \bar{u}^{(c)}(t_n + \tau), \end{aligned} \tag{19}$$

where t_n is the initialization time, so both anomalies use the climatology of the verification time. Here the average over initializations is taken after the per-forecast ratio, again as in WeatherBench 2.

The climatology $\bar{u}$ is built from ERA5 over 1990–2019 on the same 1.5° grid. Fields are first averaged by day of year and UTC hour separately, and the resulting day-of-year profile is then smoothed with WeatherBench 2's periodic triangular 61-day window. Non-leap years contribute their December 31 fields to the 366th day as well, so that every day-of-year slot is populated.

A.6 Datasets

A.6.1 Pretraining Datasets

Pretraining ran for 1,545,920 optimizer steps at a batch size of 750. The autoencoder thus saw 1.16×10^9 scalar slices of size 64×64, or about 4.7×10^{12} grid values. All downstream experiments start from the averaged weights at this final step. The slices were drawn from 217,338 completed solver runs.

The simulation server visits the 28 combinations of equation and resolution in a fixed cycle, running a configured number of simulations for each. With these settings, every combination except those of Kuramoto–Sivashinsky yields about the same number of scalar samples per cycle. Kuramoto–Sivashinsky has one state variable, whereas the other six families have three. Since every channel becomes a separate sample, it yields a third as many. The resulting mixture is close to uniform over the four resolutions, which contribute between 24.0% and 26.1% of the slices each. Across equations it is not uniform: Kuramoto–Sivashinsky supplies 5.1% of the slices, against 14.6% to 16.7% for each of the other six families.

Pretraining ran on a single machine for 16.6 days of wall-clock time, using two NVIDIA RTX A5000 GPUs with 24 GB of memory each. One GPU trained the autoencoder in bfloat16 mixed precision, and the other ran `TorchFSM` to generate the training stream.

A.6.2 PDE Dynamics Learning Data

Transonic cylinder flow (`TRA`). `TRA` is a two-dimensional dataset of compressible flow around a cylinder (Kohl et al., 2026). Its 41 trajectories were simulated with SU2 at Mach numbers from 0.50 to 0.90 and a fixed Reynolds number of 10^4. The finite-volume simulations use the Spalart–Allmaras delayed detached-eddy simulation closure. Fields from the 256×128 simulation grid are downsampled to the 128×64 Cartesian grid used for learning and evaluation. Its main characteristics are:

- evaluation resolution: 128×64;
- flow parameters: $\mathrm{Re} = 10^4$ and $\mathrm{Ma} \in [0.50, 0.90]$;

- state variables: velocity components (v_x, v_y), density ρ, and pressure p;
- interval between stored snapshots: $\Delta t = 0.002$ in nondimensional time; and
- boundary conditions: no-slip and adiabatic conditions on the cylinder surface, with far-field conditions imposed through local one-dimensional Riemann invariants.

A.7 Additional Results

A.7.1 Weather Forecasting Results Across Variables

Figure 8 extends the comparison in Figure 3 to all five forecast channels. Each of the 60 ratios is below one, so the fine-tuned model has lower RMSE for every channel, training budget, and lead time in this evaluation. The improvement is largest for `z500`, where the ratios range from 0.39 to 0.56, corresponding to RMSE reductions between 44% and 61%. Across the other four channels the ratios range from 0.53 to 0.81, i.e. RMSE reductions between 19% and 47%. The relative gain is mostly the largest for the smallest training budget in `z500`, `t850` and `q700`.

Figure 9 reports the corresponding ACC curves. The fine-tuned model has the higher ACC for every channel, training budget, and lead time, and the two curves diverge as the rollout proceeds: the from-scratch ACC decays faster with lead time in all 15 panels. The divergence is present at every training budget, and the absolute ACC of both models improves as the budget grows from one to 16 years.

Figure 10 extends the comparison against the 2D PDE foundation models in Figure 4 to all five forecast channels.

Figures 11–15 show example forecasts of the four fine-tuned models for each of the five channels, next to the ERA5 ground truth, at lead times of 6, 24, and 48 hours.

A.7.2 Six-Hour Time Differences Across ERA5 Channels

Section 3.2 weights the forecasting loss by the inverse variance of each channel's six-hour time difference. This appendix reports the statistics behind that choice.

Normalization equalizes the spread of a variable's values, not the spread of its changes. A one-step forecaster is scored on the change, so two channels that look equally scaled in the input can still contribute squared errors that differ by more than an order of magnitude. Table 10 makes the size of the effect concrete for the five pressure-level variables. We estimate the statistics from 512 transitions drawn with a fixed seed from the training interval and remapped to the $N_{\text{side}} = 64$ HEALPix grid with the same conservative scheme used in training. We express them in the standardized units seen by the model.

Table 10: Six-hour time-difference statistics for the five ERA5 pressure-level variables, in standardized model units, and the resulting channel weights w_c of Equation 2. The weights are normalized to average one and are estimated from 512 training transitions after the $N_{\text{side}} = 64$ conservative remap.

Channel	Physical σ_Δ	Unit	Standardized σ_c	w_c
`z500`	228.05	$\mathrm{m^2\,s^{-2}}$	0.0679	3.142
`t850`	1.503	K	0.0962	1.568
`q700`	$9.08 \cdot 10^{-4}$	$\mathrm{kg\,kg^{-1}}$	0.3557	0.115
`u850`	2.836	$\mathrm{m\,s^{-1}}$	0.3464	0.121
`v850`	3.238	$\mathrm{m\,s^{-1}}$	0.5170	0.054

The standardized column spans a factor of 7.6, from 0.068 for `z500` to 0.517 for `v850`. Squared, that is a factor of 58 in the per-channel contribution to a uniformly weighted mean squared error. The ordering follows atmospheric timescales: geopotential at 500 hPa is a large-scale, slowly evolving field, whereas the 850 hPa winds and mid-tropospheric humidity turn over within a few six-hour steps. Under a uniform mean, the optimizer would spend most of its capacity on the two wind channels, and a model that merely persisted `z500` would pay almost no penalty for it. Dividing

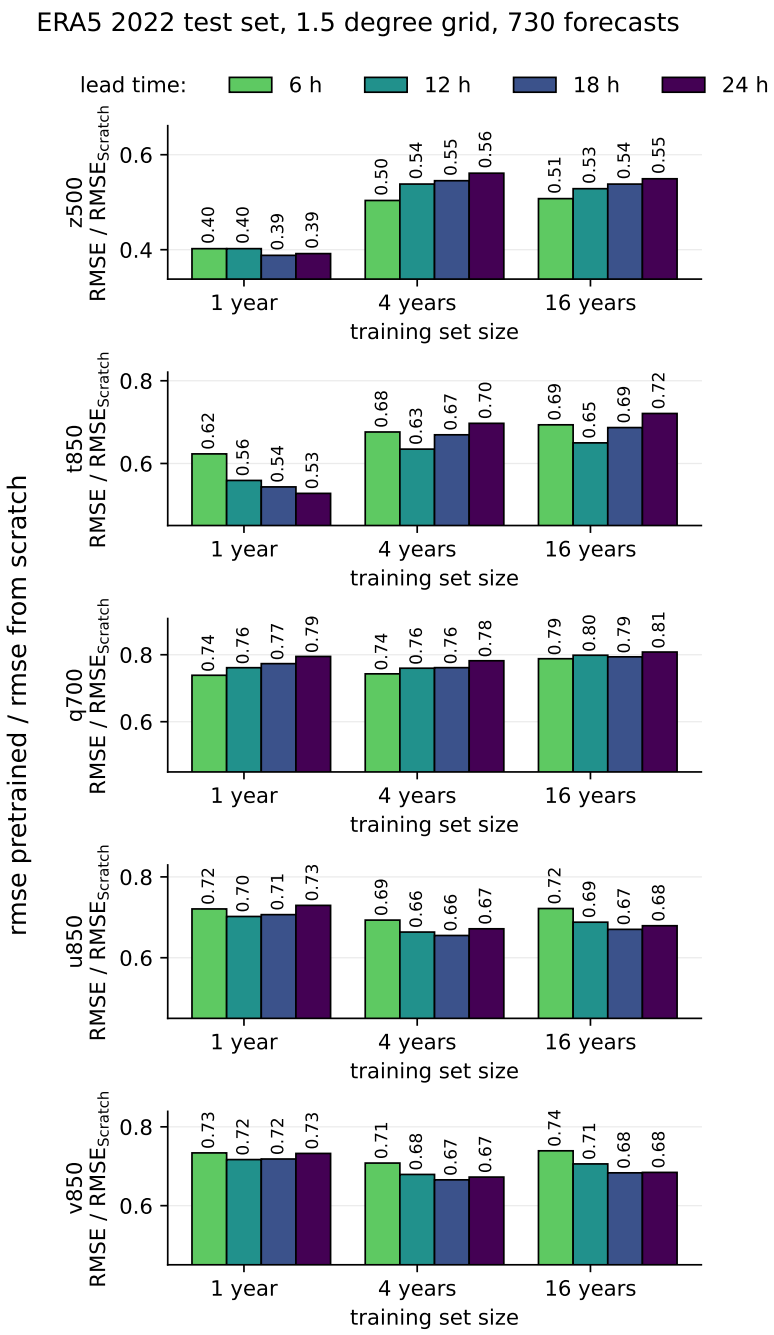


Figure 8: Ratio of RMSE of the pretrained model to the from-scratch model for the five atmospheric channels. RMSEs were aggregated over 730 forecasts initialized from the ERA5 2022 test year and conservatively remapped to the 1.5° latitude–longitude grid before calculating the error. The colours denote forecast lead times. Values below one favour the fine-tuned model.

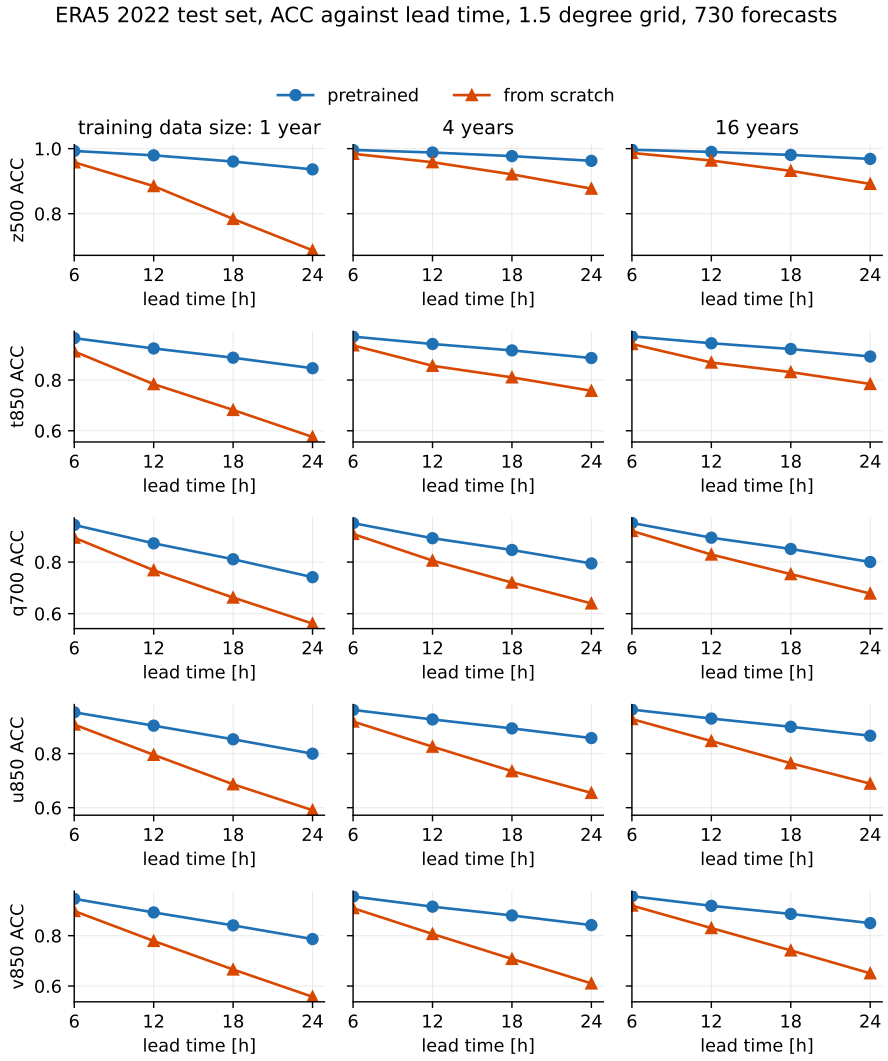


Figure 9: ACC against lead time for the fine-tuned and from-scratch models for the five atmospheric channels (rows) and the three training budgets (columns), for the same 730 forecasts and evaluation grid as Figure 8. Higher values are better. A slower decay with lead time indicates that the model retains more spatial anomaly structure during the autoregressive rollout.

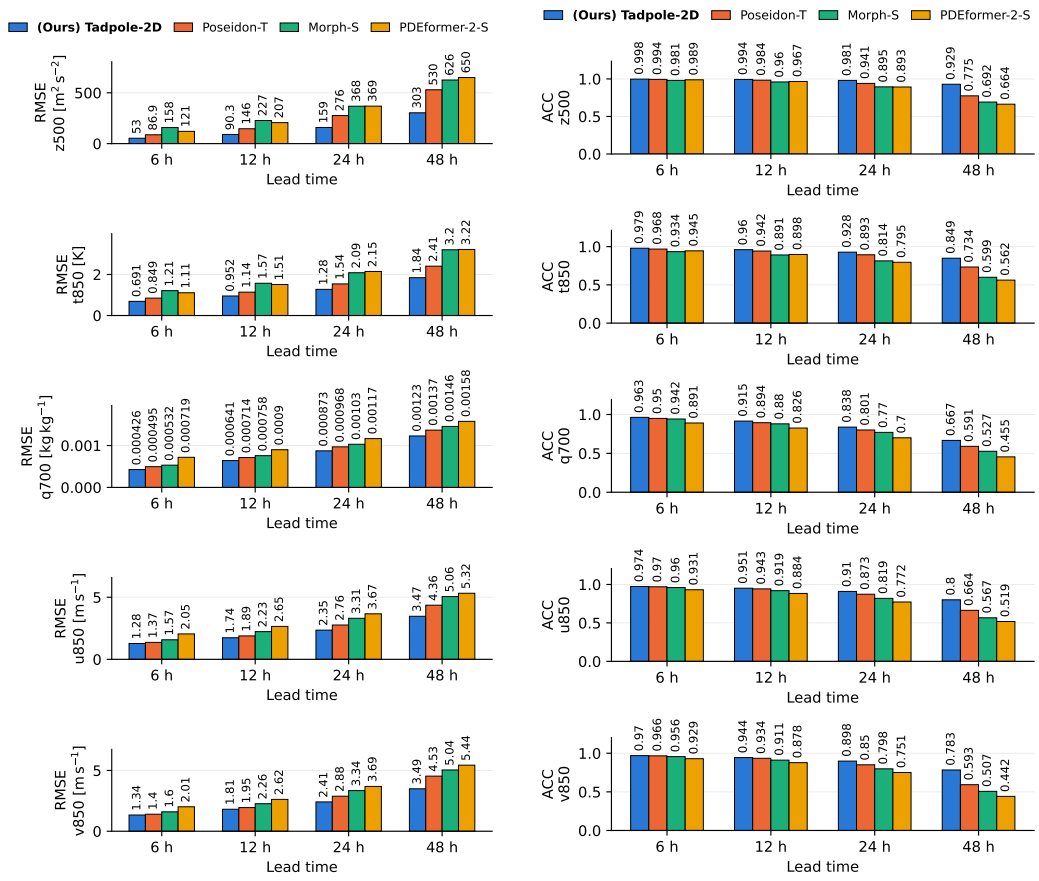


Figure 10: RMSE (left) and ACC (right) of the fine-tuned PDE foundation models across all five ERA5 channels, extending the `z500` panels of Figure 4. All models are fine-tuned on the one-step weather prediction task on the HEALPix grid. The evaluation is done using autoregressive rollout to yield 730 forecasts from the ERA5 2022 test year. Outputs are conservatively regridded from HEALPix to the $1.5°$ latitude–longitude grid.

by σ_c^2 puts every channel on a common footing: the weighted loss measures how much of each channel's own six-hour variability the model explains, rather than how many units it happens to move. Weights are rescaled to average one so that the loss magnitude, and with it the usable learning rate, is unchanged relative to the uniform objective.

The disparity is not an artefact of standardization. Figure 16 shows the six-hour persistence error, $\mathrm{mean}[(u_{t+\Delta t} - u_t)^2]$, computed over all $58{,}439$ training transitions under the min–max normalization to $[-1, 1]$ instead. The spread is about a factor of 12 between the smallest and largest channel. The ranking differs, since the two normalizations divide by different quantities, but the conclusion that no single scale serves all five channels is the same under either one.

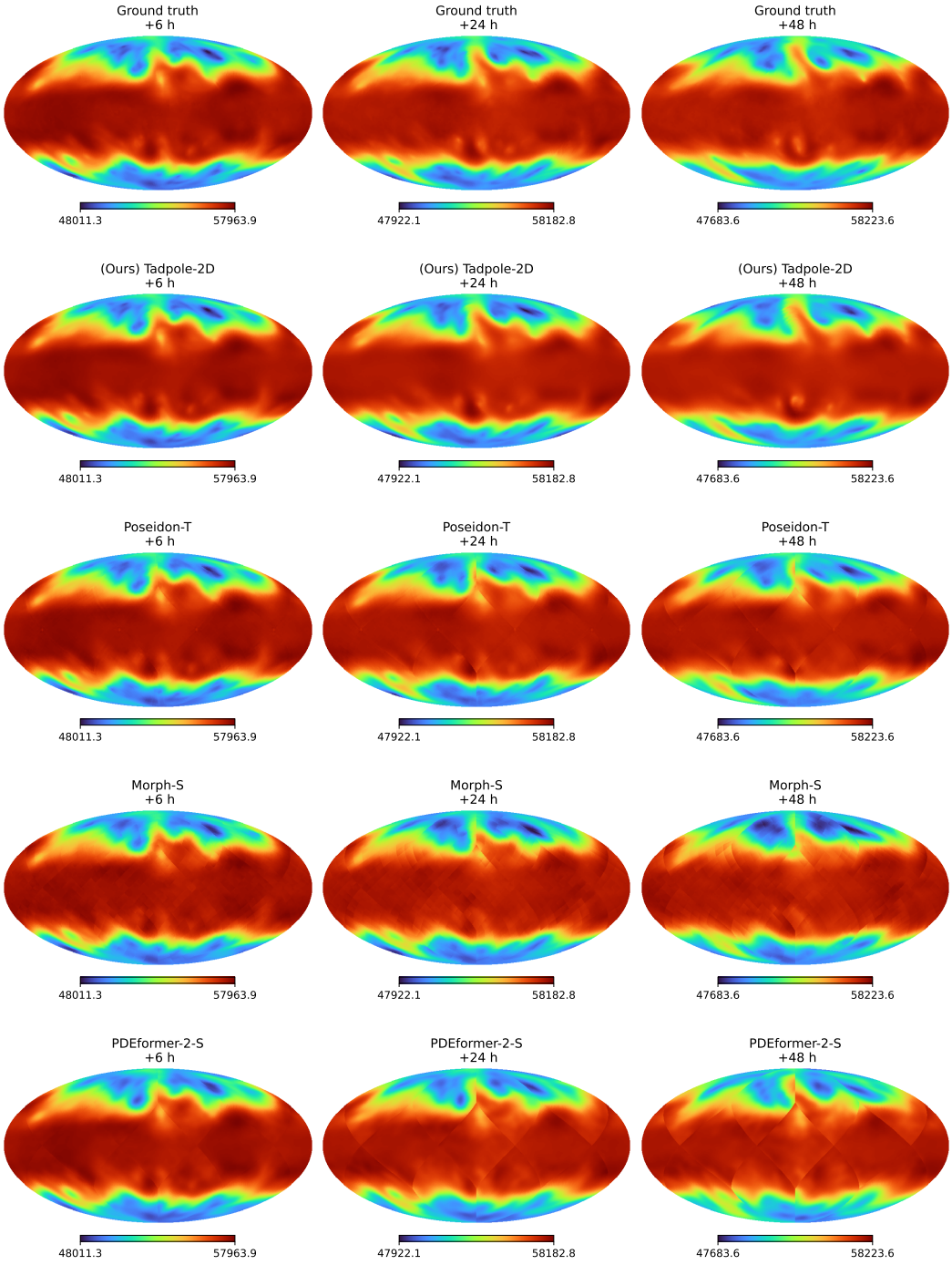


Figure 11: Forecasts of `z500` (geopotential at 500 hPa) by the four fine-tuned models, Tadpole-2D (ours), Poseidon-T, MORPH-S, and PDEformer-2-S, together with the ERA5 ground truth. The forecast is initialized from the ERA5 analysis at 00:00 UTC on 1 January 2022 and rolled out autoregressively in six-hour steps; the columns show lead times of 6, 24, and 48 hours.

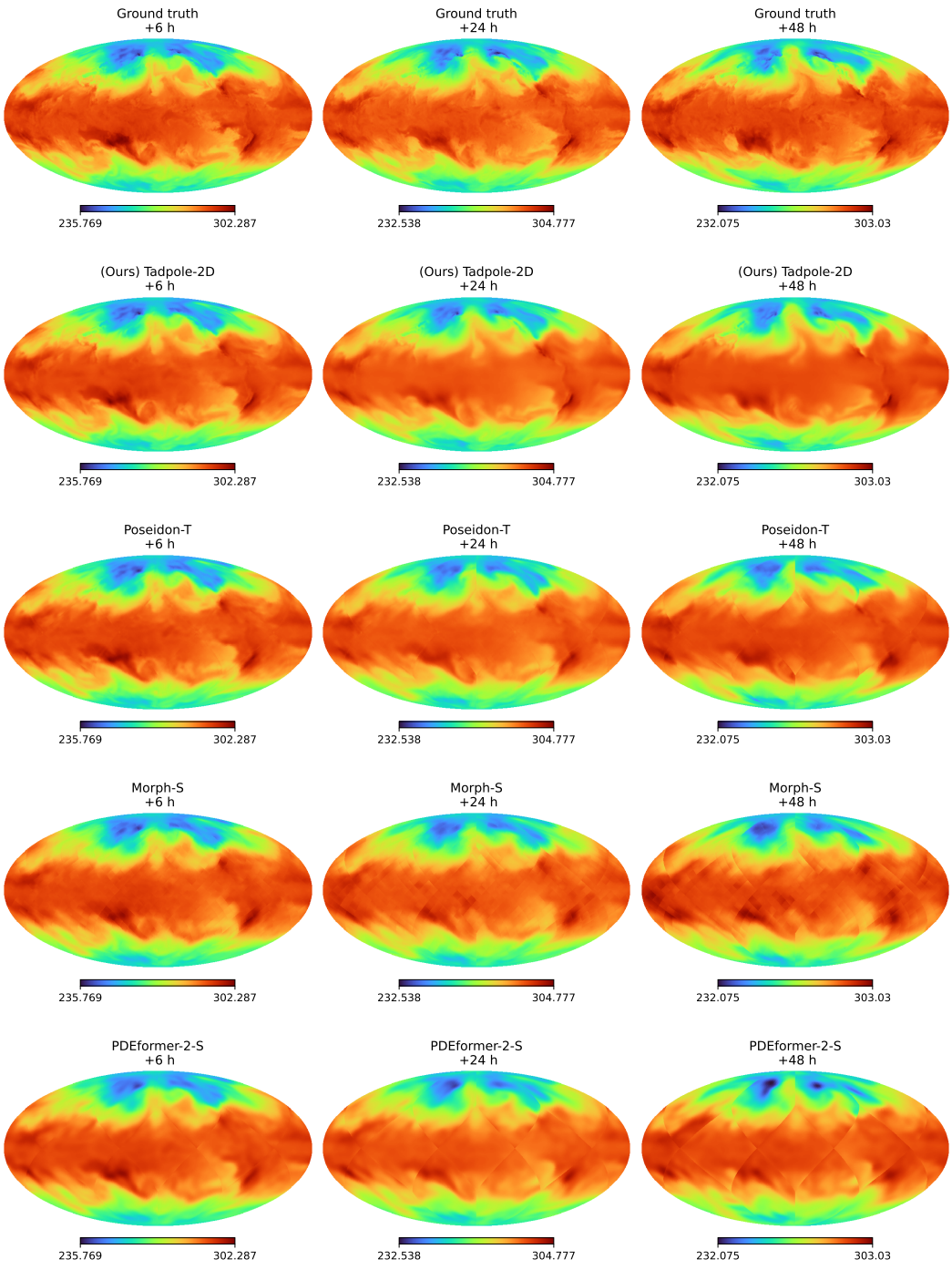


Figure 12: Forecasts of `t850` (temperature at 850 hPa) by the four fine-tuned models, together with the ERA5 ground truth, at lead times of 6, 24, and 48 hours, for the same initialization as Figure 11.

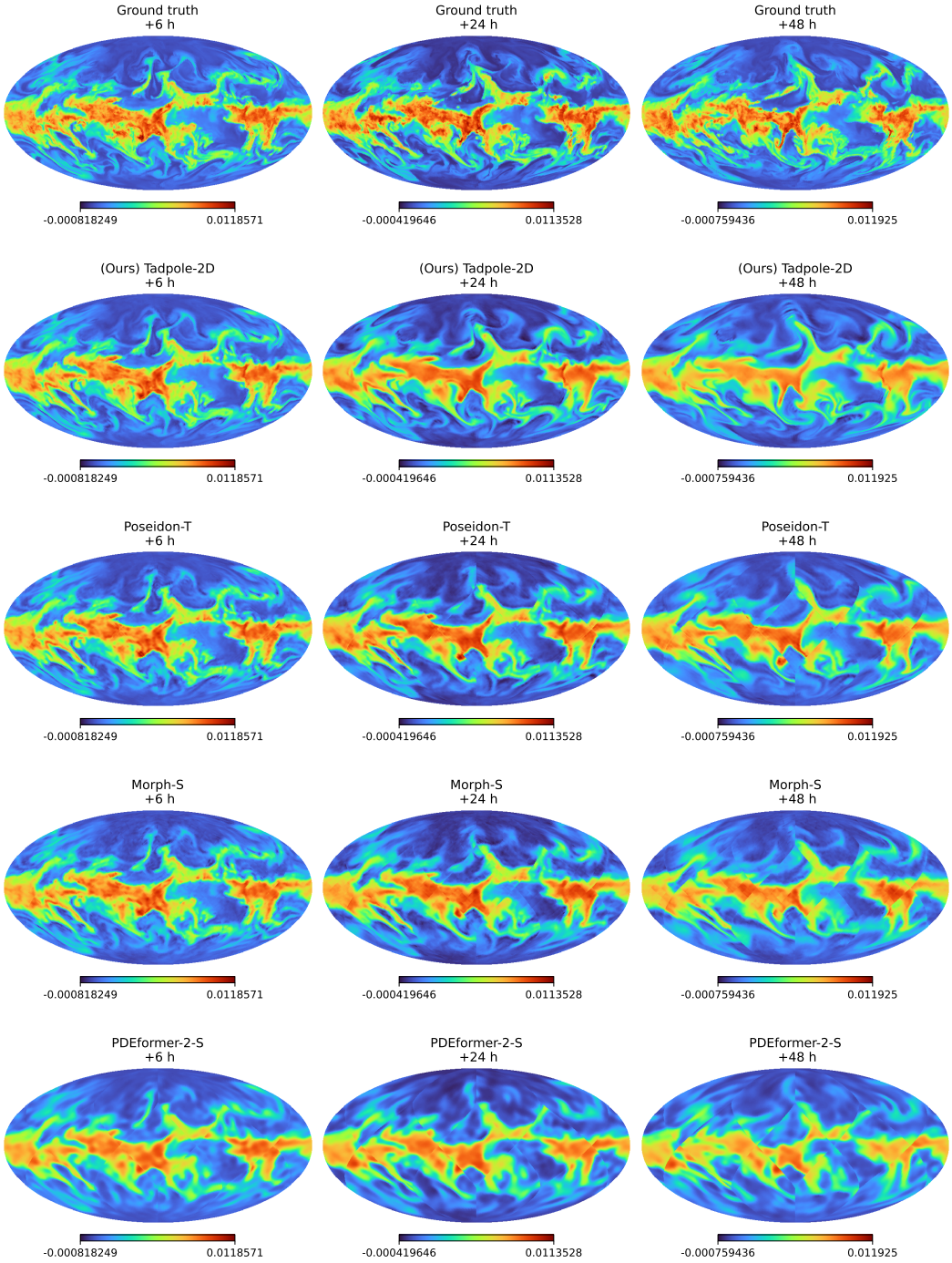


Figure 13: Forecasts of `q700` (specific humidity at 700 hPa) by the four fine-tuned models, together with the ERA5 ground truth, at lead times of 6, 24, and 48 hours, for the same initialization as Figure 11.

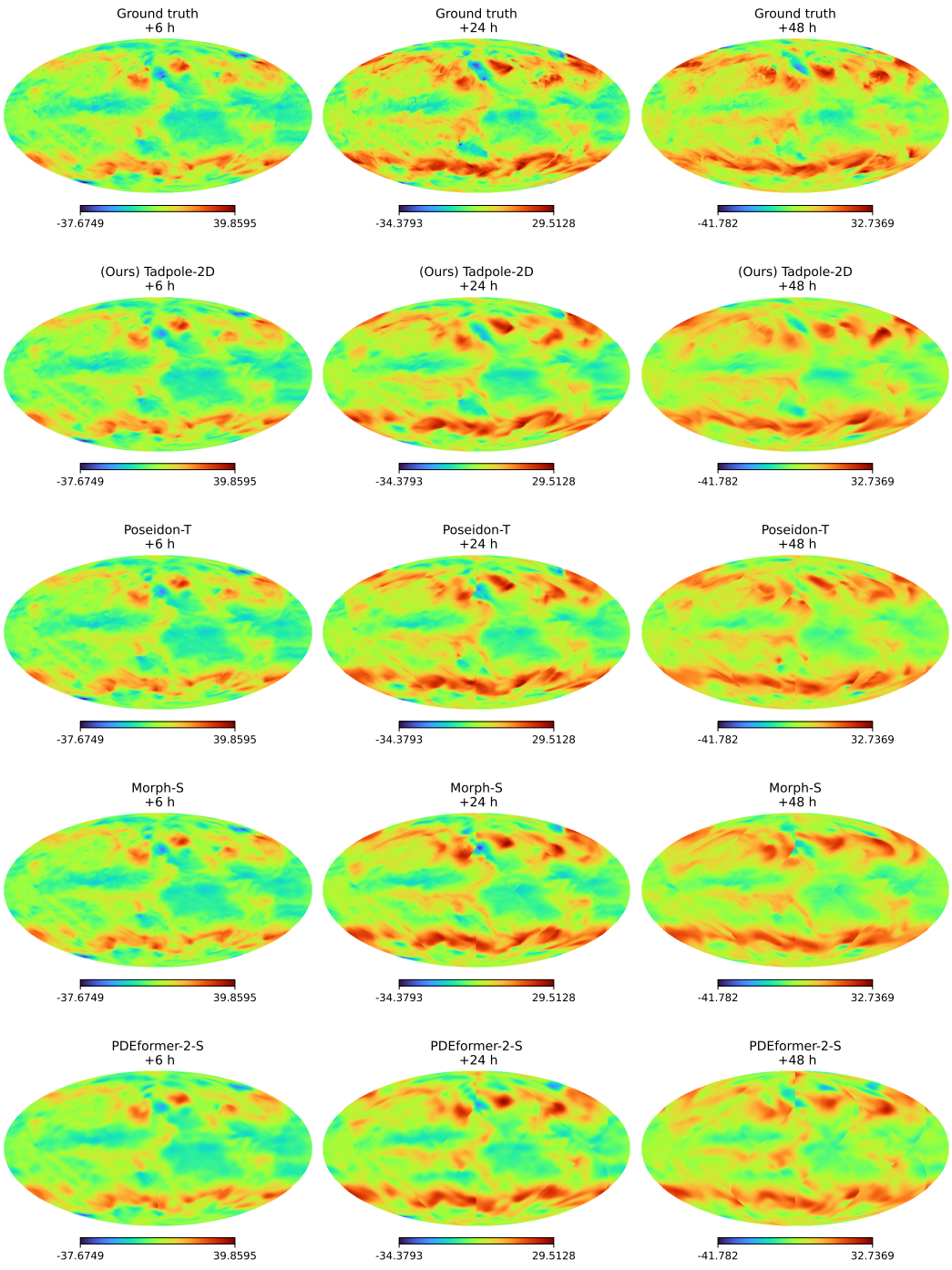


Figure 14: Forecasts of `u850` (zonal wind at 850 hPa) by the four fine-tuned models, together with the ERA5 ground truth, at lead times of 6, 24, and 48 hours, for the same initialization as Figure 11.

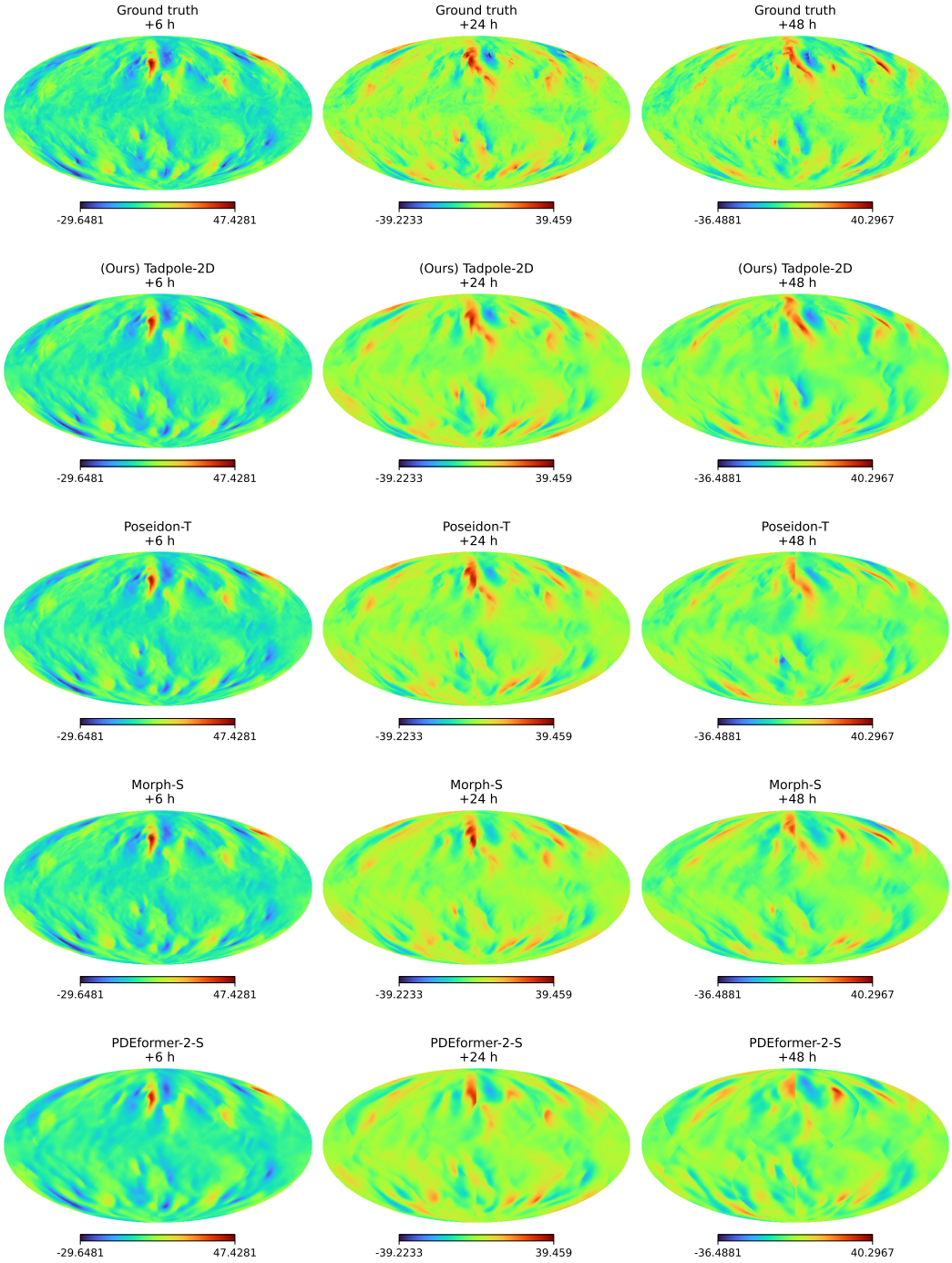


Figure 15: Forecasts of `v850` (meridional wind at 850 hPa) by the four fine-tuned models, together with the ERA5 ground truth, at lead times of 6, 24, and 48 hours, for the same initialization as Figure 11.

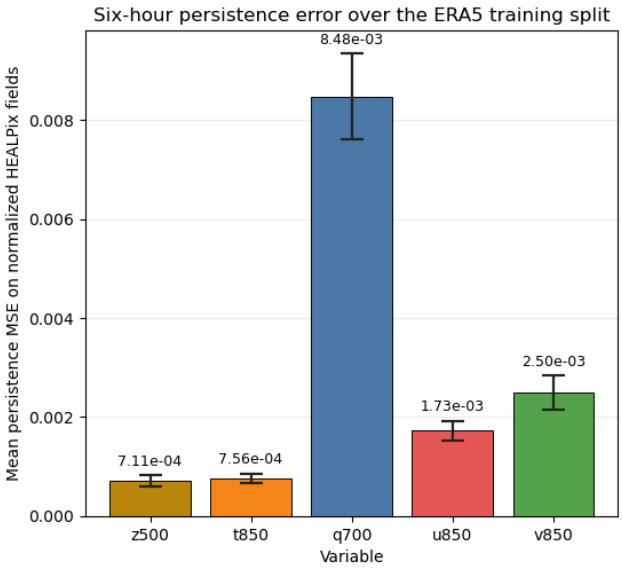


Figure 16: Six-hour persistence mean squared error per channel over the ERA5 training split, under min–max normalization to $[-1, 1]$. Capped lines show one standard deviation of the per-transition error across time, not a confidence interval for the mean. Computed at $N_{\text{side}} = 32$ on the 1.5° source grid with bilinear remapping, so the absolute values are not directly comparable with Table 10; the figure is shown for the relative spread across channels.