COMPARING AND CONTRASTING DEEP LEARNING WEATHER PREDICTION BACKBONES ON NAVIER-STOKES DYNAMICS

Matthias Karlbauer[1,2*] **Danielle C. Maddix**[1] **Abdul Fatir Ansari**[1] **Boran Han**[1]

Gaurav Gupta[1] **Yuyang Wang**[1] **Andrew Stuart**[3] **Michael W. Mahoney**[4]

[1] Amazon Web Services (AWS)
[2] Neuro-Cognitive Modeling Group, University of Tübingen, Germany
[3] Amazon Search
[4] Amazon Supply Chain Optimization Technologies

ABSTRACT

There has been remarkable progress in the development of Deep Learning Weather Prediction (DLWP) models, so much so that they are poised to become competitive with traditional numerical weather prediction (NWP) models. Indeed, a wide number of DLWP architectures—based on various backbones, including U-Net, Transformer, Graph Neural Network (GNN), or Fourier Neural Operator (FNO)—have demonstrated their potential at forecasting atmospheric states. However, due to differences in training protocols, data choices (resolution, selected prognostic variables, or additional forcing inputs), and forecast horizons, it still remains unclear which of these methods and architectures is most suitable for weather forecasting. Here, we back up and provide a detailed empirical analysis, under controlled conditions, comparing and contrasting the most prominent backbones used in DLWP models. This is done by predicting two-dimensional incompressible Navier-Stokes dynamics with different numbers of parameters and Reynolds number values. In terms of accuracy, memory consumption, and runtime, our results illustrate various tradeoffs, and they show favorable performance of FNO, in comparison with Transformer, U-Net, and GNN backbones.

1 INTRODUCTION

Deep Learning Weather Prediction (DLWP) models have recently evolved to form a promising and competitive alternative to numerical weather prediction (NWP) models (Bauer et al., 2015). Early attempts (Scher & Messori, 2018; Weyn et al., 2019) designed U-Net models (Ronneberger et al., 2015) on a cylinder mesh, learning to predict air pressure and temperature dynamics on a coarse global resolution of $5.625\,^\circ$. More recently, Pathak et al. (2022) designed FourCastNet on the basis of the Fourier Neural Operator (FNO) (Li et al., 2020b), moving to the native resolution of $0.25\,^\circ$ of the ERA5 reanalysis dataset (Hersbach et al., 2020), which amounts to 721×1440 data points covering the globe. The same dataset finds application in the Vision Transformer (ViT) (Dosovitskiy et al., 2020) based Pangu-Weather model (Bi et al., 2023) and the state-of-the-art message-passing Graph Neural Network (GNN) (Battaglia et al., 2018; Fortunato et al., 2022) based GraphCast model (Lam et al., 2022).

In a comparison of state-of-the-art DLWP models, Rasp et al. (2023) find that GraphCast generates the most accurate weather forecasts on lead-times up to ten days. GraphCast was trained on 221 variables from ERA5 (substantially more than Pangu-Weather and FourCastNet, which were trained on 67 and 24 prognostic variables, respectively), but the root of GraphCast's improved performance

*Work done during an internship at AWS.

remains entangled in details of the architecture type, choice of prognostic variables, and training protocol. Here, we seek to elucidate the effect of the model architecture, i.e., GNN, Transformer, U-Net, or FNO, and design an empirical evaluation that thoroughly compares DLWP backbone architectures under controlled conditions while freezing data choice and training protocol. To this end, we train and evaluate various architectures on two-dimensional Navier-Stokes simulations, while controlling the number of parameters to generate cost-performance tradeoff curves. We motivate our data choice of using the Navier-Stokes equations since they find application in NWP where they are used to describe parts of the atmosphere's dynamics.

With this analysis, we seek to motivate a model that has the greatest potential in addressing downsides of current DLWP models, e.g., stability of long roll-outs for climate lengthscales, uncertainty quantification, and physically meaningful predictions. Our aim is to help the community find and agree on a suitable backbone for DLWP models.

2 Related Work and Methods

We compare six model classes that form the basis for state-of-the-art DLWP models. We provide a brief overview of these methods in the following. (See Appendix A.1 for details and Table 2 in this appendix for how we modify them to vary the number of parameters.) As a naïve baseline and upper bound for our error comparison, we implement `Persistence`, which predicts the last observed state. Starting with early deep learning (DL) methods, we include convolutional long short-term memory (`ConvLSTM`) (Shi et al., 2015), which combines spatial and temporal information processing by replacing the scalar computations of LSTM gates (Hochreiter & Schmidhuber, 1997) with convolution operations. `ConvLSTM` is one of the first DL models for precipitation nowcasting and other spatiotemporal forecasting tasks, and it finds applications in Google's MetNet1 and MetNet2 (Sønderby et al., 2020; Espeholt et al., 2022). Among early DL methods, we also benchmark `U-Net`, which is one of the most prominent and versatile DL architectures, originally designed for biomedical image segmentation (Ronneberger et al., 2015) and forms the backbone of many DLWP models (Weyn et al., 2019; 2020; 2021; Karlbauer et al., 2023; Lopez-Gomez et al., 2023).

We also include two more recent architecture backbones based on Transformer and GNN, which power current state-of-the-art DLWP models. The Transformer architecture (Vaswani et al., 2017) has found success with image processing (Dosovitskiy et al., 2020) and has been applied to weather forecasting, by viewing the atmospheric state as a sequence of three-dimensional images (Gao et al., 2022). Pangu-Weather (Bi et al., 2023, by Huawei) and FuXi (Chen et al., 2023), use the `SwinTransformer` backbone (Liu et al., 2021), whereas Microsoft refines the Transformer into ClimaX, a model for weather and climate related downstream tasks (Nguyen et al., 2023). Multi-Scale MeshGraphNet (`MS MeshGraphNet`) (Fortunato et al., 2022) extends MeshGraphNet (Pfaff et al., 2020)—a message-passing GNN processing unstructured meshes—to operate on multiple grids with different resolutions. `MS MeshGraphNet` forms the basis of GraphCast (Lam et al., 2022) using a hierarchy of icosahedral meshes on the sphere.

Lastly, we benchmark architectures based on `FNO` (Li et al., 2020b). `FNO` is a type of operator learning method (Li et al., 2020a; Lu et al., 2021; Gupta et al., 2021) that learns a function-to-function mapping by combining pointwise operations in physical space and in the wavenumber/frequency domain. In contrast to the aforementioned architectures, it is a discretization invariant operator method. While `FNO` can be applied to higher resolutions than it was trained on, it may not be able to predict processes that unfold on smaller scales than observed during training (Krishnapriyan et al., 2023). These uncaptured small-scale processes can be important in turbulence modeling. We implement both a two- and a three-dimensional variant of `FNO`, as specified in Appendix A.1. We also experiment with `TFNO`, using a Tucker-based tensor decomposition (Tucker, 1966; Kolda & Bader, 2009), which is more parameter efficient. `FNO` serves as the basis for LBL's and NVIDIA's `FourCastNet` series (Pathak et al., 2022; Bonev et al., 2023; Kurth et al., 2023), which we also include in our comparison.[1]

[1]Due to the rectangular nature of our data, we consider the original FourCastNet implementation based on Guibas et al. (2021) instead of the newer Spherical Fourier Neural Operator (SFNO) (Bonev et al., 2023), which works with spherical data and is promising for weather prediction on the sphere.

Table 1: RMSE scores for experiment 1, reported for each model under different number of parameters. Errors reported in italic correspond to models that had to be retrained with gradient clipping (by norm) due to stability issues. With `OOM` and `sat`, we denote models that ran out of GPU memory and saturated, respectively. Saturated means that we did not further increase the parameters because the performance already saturated over smaller parameter ranges. Best results shown in bold.

	`#params`								
Model	5 k	50 k	500 k	1 M	2 M	4 M	8 M	16 M	32 M
`Persistence`	.5993	.5993	.5993	.5993	.5993	.5993	.5993	.5993	.5993
`ConvLSTM`	.1278	.0319	.0102	*.0090*	*.2329*	*.4443*	`OOM`	—	—
`U-Net`	.5993	.0269	.0157	.0145	.0131	.0126	.0126	`sat`	—
`FNO3D L1-8`	.3650	.2159	.1125	.1035	.1050	.0383	.0144	.0095	—
`TFNO3D L1-16`	—	—	—	.0873	.0889	.0221	.0083	.0066	.0069
`TFNO3D L4`	—	.0998	.0173	.0127	.0107	.0091	.0083	`sat`	—
`TFNO2D L4`	**.0632**	**.0139**	**.0055**	**.0046**	**.0043**	**.0054**	**.0041**	**.0046**	`sat`
`SwinTransformer`	.1637	.0603	.0107	.0084	.0070	`OOM`	—	—	—
`FourCastNet`	.1558	.0404	.0201	.0154	*.0164*	*.0153*	*.0149*	`sat`	—
`MS MeshGraphNet`	*.2559*	*.0976*	*.5209*	`OOM`	—	—	—	—	—

3 Experiments and Results

We conduct three series of experiments to explore the ability of the architectures (see Section 2) to predict the two-dimensional incompressible Navier-Stokes dynamics in a periodic domain. Our data is discretized on a two-dimensional 64×64 grid, and we design the experiments to test two levels of difficulties by generating less and more turbulent data, with Reynolds Numbers $Re = 1 \times 10^3$ (experiment 1) and $Re = 1 \times 10^4$ (experiments 2 and 3), respectively. For experiments 1 and 2, we generate 1 k samples, while experiment 3 repeats experiment 2 with an increased number, 10 k, of samples. Our experiments are designed to test: (1) easier vs. harder problems, with the modification in Re, and (2) the effect of the data size. For comparability, the initial condition and forcing of the data generation process are chosen to be identical with those in Li et al. (2020b); Gupta et al. (2021). (See Appendix A.2.) Also, following Li et al. (2020b), the models receive a context history of $h = 10$ input frames, on basis of which they autoregressively generate the remaining 40 (experiment 1) or 20 (experiments 2 and 3) frames.[2] Concretely, we apply a rolling window when generating autoregressive forecasts, by feeding the most recent h frames as input and predicting the next single frame, i.e., $\hat{y}_{t+1} = \varphi_\theta(x_{t-h,\dots,t})$, where $\hat{y}_{t+1}$ denotes the prediction of the next frame generated by model φ with trainable parameters θ, and $x_{t-h,\dots,t}$ denotes the most recent h frames provided as input concatenated along the channel dimension. The three-dimensional (T)FNO models make an exception to the autoregressive rolling window approach, by receiving the first h frames $x_{0:h}$ as input to directly generate a prediction $\hat{y}_{h+1:T}$ of the entire remaining sequence in a single step. See Appendix A.3 for our training protocol featuring hyperparameters, learning rate scheduling, and weight updates.

Experiment 1: Reynolds Number $Re = 1 \times 10^3$, 1 k samples. In this experiment, we generate less turbulent dynamics with Reynolds Number $Re = 1 \times 10^3$ and a sequence length of $T = 50$. The quantitative root mean squared error (RMSE) metric, reported in Table 1 and Figure 1 (left) shows that `TFNO2D` performs best, followed by `TFNO3D`, `SwinTransformer`, `FNO3D`, `ConvLSTM`, `U-Net`, `FourCastNet`, and `MS MeshGraphNet`. (See the qualitative results in Figure 2 in Appendix B.1 with the same findings.) All models outperform the naïve `Persistence` baseline, which always predicts the last observed state, i.e., $\hat{y}_t = x_h$. This principally indicates a successful training of all models. We observe substantial differences in the error saturation when increasing the number of parameters, which supports the ordering of architectures seen in Figure 2. Concretely, with an error of 1×10^{-2}, `MS MeshGraphNet` does not reach the accuracy level of the other models. Beyond 500 k parameters, the model hits the memory constraint and also does not con-

[2]Larger Reynolds Numbers lead to more turbulent dynamics that are harder to predict. Thus, Li et al. (2020b) selects $T = 50$ and $T = 30$ for $Re = 1e3$ and $Re = 1e4$, respectively. We follow this convention.

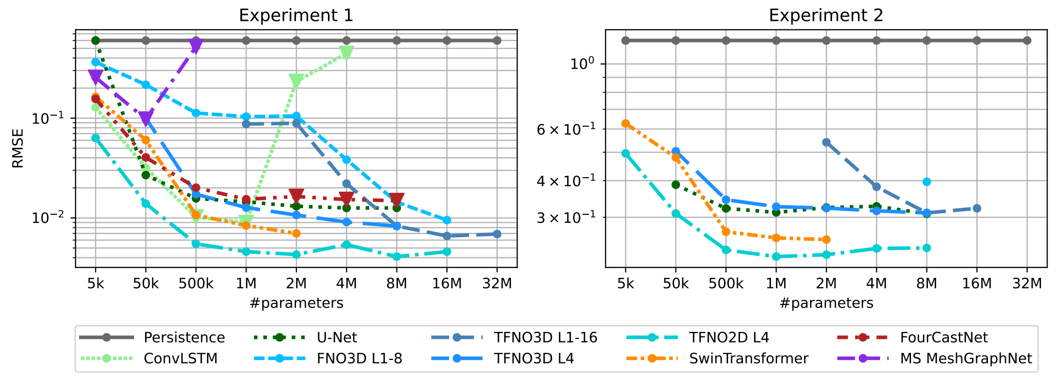

Figure 1: RMSE vs. number of parameters for models trained on Reynolds Numbers $Re = 1 \times 10^3$ (experiment 1, left) and $Re = 1 \times 10^4$ (experiment 2, right) with 1k samples. Note the different y-axis scales. Triangle markers indicate models with instability issues during training, requiring the application of gradient clipping. In the limit of growing parameters, each model converges to an individual error score (left), which seems consistent across data complexities (cf. left and right).

verge.[3] When investigating the source of this unstable training behavior, we identify remarkable effects of the graph design by comparing periodic 4-stencil, 8-stencil, and Delaunay triangulation graphs, where the latter supports a stable convergence most (see Figure 3 in Appendix B.1 for details). Throughout our experiments, we use the 4-stencil graph. `ConvLSTM` is competitive within the low-parameter-regime, saturating around an RMSE of 9×10^{-3}, and becomes unstable with large channel sizes (which we could not compensate even with gradient clipping), runs out of memory beyond $4\,\mathrm{M}$ parameters, and suffers from exponential runtime complexity (see Figure 4 (right) in Appendix B.1). Similarly, `SwinTransformer` generates comparably accurate predictions, reaching an error of 7×10^{-3}, before quickly running out of memory when going beyond $2\,\mathrm{M}$ parameters. `U-Net` and `FourCastNet` express a similar behavior, saturating at the $1\,\mathrm{M}$ parameter configuration and reaching error levels of 1.2×10^{-2} and 1.5×10^{-2}, respectively. In `FNO3D` and `TFNO3D`, we observe a two-staged saturation, where the models first converge to a poor error regime of 1×10^{-1}, albeit approaching a remarkably smaller RMSE of 9×10^{-3} and 6×10^{-3}, respectively, when increasing the number of layers from 1 at `#params` $\leq 2\,\mathrm{M}$ to 2, 4, 8, and 16 to obtain the respective larger parameter counts.[4] Instead, when fixing the numbers of layers at $l = 4$ and varying the number of channels in `TFNO3D L4`, we observe better performance compared to the single-layer `TFNO3D L1-16` in the low-parameter regime (until $2\,\mathrm{M}$ parameters), albeit not competitive with other models. To additionally explore the effect of the number of layers vs. channels in `TFNO3D`, we vary the number of parameters either by increasing the layers over $l \in [1, 2, 4, 8, 16]$, while fixing the number of channels at $c = 32$ in `TFNO3D L1-16`, or by increasing the number of channels over $c \in [2, 8, 11, 16, 22, 32]$ while fixing the number of layers at $l = 4$ in `TFNO3D 4L`. Consistent with Li et al. (2020b), we observe the performance saturating at four layers. Finally, the autoregressive `TFNO2D` performs remarkably well across all parameter ranges—converging at an unparalleled RMSE score of 4×10^{-3}—while, at the same time, constituting a reasonable trade-off between memory consumption and runtime complexity (see Figure 4 in Appendix B.1). From this we conclude that, at least for periodic fluid flow simulation, `FNO2D` marks a promising choice, suggesting its application to real-world weather forecasting scenarios.

Experiment 2: Reynolds Number $Re = 1 \times 10^4$, 1 k samples. In this experiment, to stress test the consistency of the model order found in experiment 1, we generate more turbulent data by increasing the Reynolds Number Re by an order of magnitude and reducing the simulation time and sequence length to $T = 30$ timesteps. To reduce our search over various architectures, we

[3]Experiments are performed on an AWS `g5.12xlarge` instance featuring four NVIDIA A10G GPUs with 23 GB RAM each. We used single GPU training throughout all our experiments.

[4]We observe a similar behavior (not shown) when experimenting with the number of blocks vs. layers in `SwinTransformer`, suggesting to prioritise more layers per block over more blocks with less layers.

focus here on stable models only by removing those that depend on gradient clipping. We make the same observations as in experiment 1, confirming `TFNO2D` as the most accurate model, followed by `SwinTransformer`, `TFNO3D`, and `U-Net` on this harder task. See Figure 1 (right) for quantitative results and Figure 6 in Appendix B.2 for qualitative results.

Experiment 3: Reynolds Number $Re = 1 \times 10^4$, 10 k samples. Since in experiment 2, the three-dimensional TFNO models with `#params` $\geq 8\,\mathrm{M}$ start to show a slight tendency to overfit (not shown), we repeat the experiment, increasing the number of training samples by an order of magnitude to $10\,\mathrm{k}$. Figure 5, Figure 7 and Table 4 in Appendix B.2 show that the same findings hold on experiment 3, which has a larger number of $10\,\mathrm{k}$ samples.

4 Discussion

In this work, we obtain insights into which DLWP models are more suitable for weather forecasting by devising controlled experiments, which are useful more generally, to compare these models. In particular, we fix the input data and training protocol and vary the architecture and number of parameters. In this limited setup on the synthetic two-dimensional periodic incompressible Navier-Stokes dataset, we find that `TFNO2D` performs the best at predicting the dynamics, followed by `TFNO3D`, `SwinTransformer`, `FNO3D`, `ConvLSTM`, `U-Net`, `FourCastNet`, and `MS MeshGraphNet`. Importantly, throughout our experiments, we see that all benchmarked models eventually saturate at an individual error regime, indicating that scaling laws could not be determined for the models and task at hand. We thus identify a value for further research in developing an architecture that scales with increased parameters without converging at a specific error level. The spherical Fourier Neural Operator SFNO (Bonev et al., 2023) marks a reasonable starting point for further exploration, ranging from increasing the number of prognostic variables, to providing more context to the models by means of forcing fields, to incorporating physical knowledge (Li et al., 2022; Hansen et al., 2023) and moving beyond deterministic forecasts (Cachay et al., 2023; Gao et al., 2023; Price et al., 2023).

A limitation of our work is the sole consideration of synthetic data. We seek to focus our future endeavors on repeating the analysis presented here on WeatherBench (Rasp et al., 2020) to verify our claims and suggestions on real-world weather data that is represented on the sphere. In addition, although we enable circular padding in the compared architectures, the periodic nature of the data we use may favor the inductive bias of FNO. The periodicity exhibited in our data matches that of weather dynamics on global scale. Throughout our experiments, we perform only single model runs. When experimenting with different seeds, we do not encounter severe variations, which is also supported by the consistent RMSE vs. parameter plots. We note the poor results for `MS MeshGraphNet`, which we attribute to the underlying graph and to our training protocol. Seeing that GraphCast has been trained very carefully on 221 prognostic variables, its lead over other DLWP may be dominated by the number of variables and by the training protocol, and less to the actual architecture. This hypothesis, however, needs further investigations in future work such as on WeatherBench.

References

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018. 1

Peter Bauer, Alan Thorpe, and Gilbert Brunet. The quiet revolution of numerical weather prediction. *Nature*, 525(7567):47–55, 2015. 1

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(7970):533–538, 2023. 1, 2

Boris Bonev, Thorsten Kurth, Christian Hundt, Jaideep Pathak, Maximilian Baust, Karthik Kashinath, and Anima Anandkumar. Spherical fourier neural operators: Learning stable dynamics on the sphere. *arXiv preprint arXiv:2306.03838*, 2023. 2, 5

Salva Rühling Cachay, Bo Zhao, Hailey James, and Rose Yu. Dyffusion: A dynamics-informed diffusion model for spatiotemporal forecasting. *arXiv preprint arXiv:2306.01984*, 2023. 5

Lei Chen, Xiaohui Zhong, Feng Zhang, Yuan Cheng, Yinghui Xu, Yuan Qi, and Hao Li. Fuxi: A cascade machine learning forecasting system for 15-day global weather forecast. *arXiv preprint arXiv:2306.12873*, 2023. 2

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020. 1, 2

Lasse Espeholt, Shreya Agrawal, Casper Sønderby, Manoj Kumar, Jonathan Heek, Carla Bromberg, Cenk Gazen, Rob Carver, Marcin Andrychowicz, Jason Hickey, et al. Deep learning for twelve hour precipitation forecasts. *Nature communications*, 13(1):1–10, 2022. 2

Meire Fortunato, Tobias Pfaff, Peter Wirnsberger, Alexander Pritzel, and Peter Battaglia. Multiscale meshgraphnets. In *ICML 2022 2nd AI for Science Workshop*, 2022. 1, 2, 10, 13

Kunihiko Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological cybernetics*, 20(3-4):121–136, 1975. 10

Zhihan Gao, Xingjian Shi, Hao Wang, Yi Zhu, Yuyang Wang, Mu Li, and Dit-Yan Yeung. Earthformer: Exploring space-time transformers for earth system forecasting. In *Advances in Neural Information Processing Systems*, volume 35, pp. 25390–25403, 2022. 2

Zhihan Gao, Xingjian Shi, Boran Han, Hao Wang, Xiaoyong Jin, Danielle C. Maddix, Yi Zhu, Mu Li, and Yuyang Wang. PreDiff: Precipitation nowcasting with latent diffusion models. In *Advances in Neural Information Processing Systems*, 2023. 5

John Guibas, Morteza Mardani, Zongyi Li, Andrew Tao, Anima Anandkumar, and Bryan Catanzaro. Adaptive fourier neural operators: Efficient token mixers for transformers. *arXiv preprint arXiv:2111.13587*, 2021. 2

Gaurav Gupta, Xiongye Xiao, and Paul Bogdan. Multiwavelet-based operator learning for differential equations. In *Advances in Neural Information Processing Systems*, volume 34, pp. 24048–24062, 2021. 2, 3, 10

Derek Hansen, Danielle C. Maddix, Shima Alizadeh, Gaurav Gupta, and Michael W. Mahoney. Learning physical models that can respect conservation laws. In *International Conference on Machine Learning*, volume 202, pp. 12469–12510. PMLR, 2023. 5

Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, et al. The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020. 1

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997. 2

Matthias Karlbauer, Nathaniel Cresswell-Clay, Dale R Durran, Raul A Moreno, Thorsten Kurth, and Martin V Butz. Advancing parsimonious deep learning weather prediction using the healpix mes. *Authorea Preprints*, 2023. 2

Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3): 455–500, 2009. 2

Aditi S Krishnapriyan, Alejandro F Queiruga, N Benjamin Erichson, and Michael W Mahoney. Learning continuous models for continuous physics. *Communications Physics*, 6(1):319, 2023. 2

Thorsten Kurth, Shashank Subramanian, Peter Harrington, Jaideep Pathak, Morteza Mardani, David Hall, Andrea Miele, Karthik Kashinath, and Anima Anandkumar. Fourcastnet: Accelerating global high-resolution weather forecasting using adaptive fourier neural operators. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, pp. 1–11, 2023. 2

Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. Graphcast: Learning skillful medium-range global weather forecasting. *arXiv preprint arXiv:2212.12794*, 2022. 1, 2

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6755–6766, 2020a. 2

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar, et al. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2020b. 1, 2, 3, 4, 10

Zongyi Li, Miguel Liu-Schiaffini, Nikola Kovachki, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Learning dissipative dynamics in chaotic systems. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16768–16781, 2022. 5

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021. 2, 10

Ignacio Lopez-Gomez, Amy McGovern, Shreya Agrawal, and Jason Hickey. Global extreme heat forecasting using neural weather models. *Artificial Intelligence for the Earth Systems*, 2(1): e220035, 2023. 2

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021. 2

Tung Nguyen, Johannes Brandstetter, Ashish Kapoor, Jayesh K Gupta, and Aditya Grover. Climax: A foundation model for weather and climate. *arXiv preprint arXiv:2301.10343*, 2023. 2

Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022. 1, 2

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2020. 2, 13

Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Timo Ewalds, Andrew El-Kadi, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, Remi Lam, and Matthew Willson. Gencast: Diffusion-based ensemble forecasting for medium-range weather. *arXiv preprint arXiv:2312.15796*, 2023. 5

Stephan Rasp, Peter D Dueben, Sebastian Scher, Jonathan A Weyn, Soukayna Mouatadid, and Nils Thuerey. Weatherbench: a benchmark data set for data-driven weather forecasting. *Journal of Advances in Modeling Earth Systems*, 12(11):e2020MS002203, 2020. 5

Stephan Rasp, Stephan Hoyer, Alexander Merose, Ian Langmore, Peter Battaglia, Tyler Russel, Alvaro Sanchez-Gonzalez, Vivian Yang, Rob Carver, Shreya Agrawal, et al. Weatherbench 2: A benchmark for the next generation of data-driven global weather models. *arXiv preprint arXiv:2308.15560*, 2023. 1

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241. Springer, 2015. 1, 2

Sebastian Scher and Gabriele Messori. Predicting weather forecast uncertainty with machine learning. *Quarterly Journal of the Royal Meteorological Society*, 144(717):2830–2841, 2018. 1

Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, volume 28, 2015. 2

Casper Kaae Sønderby, Lasse Espeholt, Jonathan Heek, Mostafa Dehghani, Avital Oliver, Tim Salimans, Shreya Agrawal, Jason Hickey, and Nal Kalchbrenner. Metnet: A neural weather model for precipitation forecasting. *arXiv preprint arXiv:2003.12140*, 2020. 2

Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3): 279–311, 1966. 2

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. 2

Jonathan A Weyn, Dale R Durran, and Rich Caruana. Can machines learn to predict weather? using deep learning to predict gridded 500-hpa geopotential height from historical weather data. *Journal of Advances in Modeling Earth Systems*, 11(8):2680–2693, 2019. 1, 2

Jonathan A Weyn, Dale R Durran, and Rich Caruana. Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. *Journal of Advances in Modeling Earth Systems*, 12(9):e2020MS002109, 2020. 2

Jonathan A Weyn, Dale R Durran, Rich Caruana, and Nathaniel Cresswell-Clay. Sub-seasonal forecasting with a large ensemble of deep-learning weather prediction models. *Journal of Advances in Modeling Earth Systems*, 13(7):e2021MS002502, 2021. 2

A Model, Data, and Training Specifications

In this section, we discuss the model configurations and how we vary the number of parameters in our experiments. In addition, we detail the dataset generation and training protocols.

A.1 Model Configurations

We compare six model classes that form the basis for state-of-the-art DLWP models. We provide details about each model and how we modify them in order to vary the number of parameters below. Table 2 provides an overview and summary of the parameters and model configurations.

ConvLSTM We first implement an encoder—to increase the model's receptive field—consisting of three convolutions with kernel size $k = 3$, stride $s = 1$, padding $p = 1$, set $\mathrm{padding\_mode} = \mathrm{circular}$ to match the periodic nature of our data, and implement $\tanh$ activation functions. We add four `ConvLSTM` cells, also with circular padding and varying channel depth (see Table 2 for details), followed by a linear output layer. Being the only recurrent model, we

Table 2: Model configurations partitioned by model and number of parameters (which amount to the trainable weights). For configurations that are not specified here, the default settings from the respective model config files are applied, e.g., `ConvLSTM` employs the default from `configs/model/convlstm.yaml`, while overriding `hidden_sizes` by the content of the "Dim." column of this table. Details are also reported in the respective model paragraphs of Appendix A.1.

Model	`#params`	Model-specific configurations			
		Enc.	Dim.	Dec.	
`ConvLSTM`	5 k	3 × `Conv2D` with `tanh()`	4×4	1 × `Conv2D`	
	50 k		4×13		
	500 k		4×40		
	1 M		4×57		
	2 M		4×81		
	4 M		4×114		
	8 M	—	—	—	
		#modes	Dim.	#layers	
`(T)FNO3D L1-16`	5 k	3×3	11	1	
	50 k	3×3	32	1	
	500 k	3×7	32	1	
	1 M	3×10	32	1	
	2 M	3×12	32	1	
	4 M	3×12	32	2	
	8 M	3×12	32	4	
	16 M	3×12	32	8	
	32 M	3×12	32	16	
		#modes	Dim.	#layers	
`TFNO3D L4`	5 k	—	—	—	
	50 k	3×12	8	4	
	500 k	3×12	11	4	
	1 M	3×12	16	4	
	2 M	3×12	22	4	
	4 M	3×12	32	4	
	8 M	3×12	45	4	
		#heads	Dim.	#blocks	#lrs/blck
`Swin-Transformer`	5 k	1	8	1	1
	50 k	2	8	2	2
	500 k	4	40	2	4
	1 M	4	60	2	4
	2 M	4	88	2	4

Model	Model-specific configurations		
	Dim. (hidden sizes)		
`U-Net`	$[1, 2, 4, 8, 8]$		
	$[3, 6, 12, 24, 48]$		
	$[8, 16, 32, 64, 128]$		
	$[12, 24, 48, 96, 192]$		
	$[16, 32, 64, 128, 256]$		
	$[23, 46, 92, 184, 368]$		
	$[33, 66, 132, 264, 528]$		
	#modes	Dim.	#layers
`TFNO2D L4`	2×12	2	4
	2×12	8	4
	2×12	27	4
	2×12	38	4
	2×12	54	4
	2×12	77	4
	2×12	108	4
	2×12	154	4
	—	—	—
	Dim.	#layers	
`FourCastNet`	12	1	
	64	1	
	112	4	
	160	4	
	232	4	
	326	4	
	468	4	
	$D_{\text{processor}}$	D_{other}	
`MS Mesh-GraphNet`	8	8	
	34	32	
	116	32	
	—	—	
	—	—	

perform ten steps of teacher forcing before switching to closed loop to autoregressively unroll a prediction into the future.

U-Net We implement a five-layer encoder-decoder architecture with avgpool and transposed convolution operations for down and up-sampling, respectively. On each layer, we employ two consecutive convolutions with ReLU activations (Fukushima, 1975) and apply the same parameters described above in the encoder for `ConvLSTM`. See Table 2 for the numbers of channels hyperparameter setting.

SwinTransformer Enabling circular padding and setting patch size $p = 2$, we benchmark the shifted window transformer (Liu et al., 2021) by varying the number of channels, heads, layers, and blocks, as detailed in Table 2, while keeping remaining parameters at their defaults.

MS MeshGraphNet We formulate a periodically connected graph to apply Multi-Scale MeshGraphNet (`MS MeshGraphNet`) with two stages, featuring 1-hop and 2-hop neighborhoods, and follow Fortunato et al. (2022) by encoding the distance and angle to neighbors in the edges. We employ four processor and two node/edge encoding and decoding layers and set $\text{hidden\_dim} = 32$ for processor, node encoder, and edge encoder, unless overridden (see Table 2).

FNO We compare three variants of `FNO`: Two three-dimensional formulations, which process the temporal and both spatial dimensions simultaneously to generate a three-dimensional output of shape $[T, H, W]$ in one call, and a two-dimensional version, which only operates on the spatial dimensions of the input and autoregressively unrolls a prediction into the future. While fixing the lifting and projection channels at 256, we vary the number of Fourier modes, channel depth, and number of layers according to Table 2.

FourCastNet We choose a patch size of $p = 4$, fix $\text{num\_blocks} = 4$, enable periodic padding in both spatial dimensions, and keep the remaining parameters at their default values while varying the number of layers and channels as specified in Table 2.

A.2 Data Generation

We provide additional information about the data generation process in Table 3, which we keep as close as possible to that reported in Li et al. (2020b) and Gupta et al. (2021).

A.3 Training protocol

In the experiments, we use the Adam optimizer with learning rate $\eta = 1 \times 10^{-3}$ (except for `MS MeshGraphNet`, which only converged with a smaller learning rate of $\eta = 1 \times 10^{-4}$) and cosine learning rate scheduling to train all models with a batch size of $B = 4$, effectively realizing $125\,\text{k}$

Table 3: Settings for training, validation, and test data generation in the experiments, where f, T, δ_t, and ν denote the dynamic forcing, sequence length (corresponding to the simulation time, which, in our case, matches the number of frames, i.e., $\Delta t = 1$), time step size for the simulation, and viscosity (which is the inverse of the Reynolds Number, i.e., $Re = 1/\nu$), respectively. The parameters α and τ parameterize the Gaussian random field to sample an initial condition (IC) resembling the first timestep.

	Simulation parameters				IC		`#samples`		
Experiment	f	T	δ_t	ν	α	τ	Train	Val.	Test
1	*	50	1×10^{-2}	1×10^{-3}	2.5	7	1000	50	200
2	*	30	1×10^{-4}	1×10^{-4}	2.5	7	1000	50	200
3	*	30	1×10^{-4}	1×10^{-4}	2.5	7	10000	50	200

* $f = 0.1(\sin(2\pi(x + y)) + \cos(2\pi(x + y)))$, with $x, y \in [0, 1, \ldots, 63]$.

weight update steps, relating to 500 and 50 epochs, respectively, for $1\,\mathrm{k}$ and $10\,\mathrm{k}$ samples.[5] For the training objective and loss function, we choose the mean squared error (MSE) between the model outputs and respective ground truth frames, that is $\mathcal{L} = \mathrm{MSE}(\hat{y}_{h+1:T}, y_{h+1:T})$. Note that, to stabilize training, we have to employ gradient clipping (by norm) for selected models, indicated by italic numbers in tables and triangle markers in figures.

B Additional Results and Materials

In this section, we provide additional empirical results for the three experiments.

B.1 Results from Experiment 1: Reynolds Number $Re = 1 \times 10^3$

Figure 2 illustrates the initial and end conditions along with the respective predictions of all models. Qualitatively, we find there exist parameter settings for all models to successfully unroll a plausible prediction of the Navier-Stokes dynamics over 40 frames into the future, as showcased by the last predicted frame, i.e., $\hat{y}_{t=T}$ (see the third and fifth row of Figure 2). When computing the difference between the prediction and ground truth, i.e., $d = \hat{y} - y$, we observe clear variations in the accuracy of the model outputs, denoted by the saturation of the difference plots in the second and fourth row of Figure 2. Interestingly, this difference plot also reveals artifacts in the outputs of selected models: `SwinTransformer` and `FourCastNet` generate undesired patterns that resemble their windowing and patching mechanisms, whereas the 2-hop neighborhood, which was chosen as the resolution of the coarser grid, is baked into the output of `MS MeshGraphNet`. According to the

Table 4: RMSE scores partitioned by experiments and reported for each model under different numbers of parameters. Errors reported in italic correspond to models that had to be retrained with gradient clipping (by norm) due to stability issues. With `OOM` and `sat`, we denote models that ran out of GPU memory and saturated, meaning that we did not train models with more parameters because the performance already saturated over smaller parameter ranges. Best results are reported in bold. More details about architecture specifications are reported in Appendix A.1 and Table 2.

		#params								
	Model	5 k	50 k	500 k	1 M	2 M	4 M	8 M	16 M	32 M
Experiment 1	Persistence	.5993	.5993	.5993	.5993	.5993	.5993	.5993	.5993	.5993
	ConvLSTM	.1278	.0319	.0102	*.0090*	*.2329*	*.4443*	OOM	—	—
	U-Net	.5993	.0269	.0157	.0145	.0131	.0126	.0126	sat	—
	FNO3D L1-8	.3650	.2159	.1125	.1035	.1050	.0383	.0144	.0095	—
	TFNO3D L1-16	—	—	—	.0873	.0889	.0221	.0083	.0066	.0069
	TFNO3D L4	—	.0998	.0173	.0127	.0107	.0091	.0083	sat	—
	TFNO2D L4	**.0632**	**.0139**	**.0055**	**.0046**	**.0043**	**.0054**	**.0041**	**.0046**	sat
	SwinTransformer	.1637	.0603	.0107	.0084	.0070	OOM	—	—	—
	FourCastNet	.1558	.0404	.0201	.0154	*.0164*	*.0153*	*.0149*	sat	—
	MS MeshGraphNet	*.2559*	*.0976*	*.5209*	OOM	—	—	—	—	—
Experiment 2	Persistence	1.202	1.202	1.202	1.202	1.202	1.202	1.202	1.202	1.202
	U-Net	—	.3874	.3217	.3117	.3239	.3085	sat	—	—
	TFNO3D L1-8	—	—	—	—	.5407	.3811	.3105	.3219	sat
	TFNO3D L4	—	.5038	.3444	.3261	.3224	.3155	.3105	sat	—
	TFNO2D L4	**.4955**	**.3091**	**.2322**	**.2322**	**.2236**	**.2349**	**.2358**	sat	—
	SwinTransformer	.6266	.4799	.2678	.2552	.2518	OOM	—	—	—
Experiment 3	Persistence	1.202	1.202	1.202	1.202	1.202	1.202	1.202	1.202	1.202
	U-Net	—	.3837	.3681	.2497	.3162	.2350	.2383	sat	—
	TFNO3D L1-16	—	—	—	—	.5146	.2805	.1814	.1570	.1709
	TFNO3D L4	—	.4799	.2754	.2438	.2197	.2028	.1814	.1740	sat
	TFNO2D L4	**.4846**	**.2897**	**.1778**	**.1585**	**.1449**	**.1322**	**.1248**	**.1210**	sat
	SwinTransformer	.6187	.4698	.2374	.2078	.1910	OOM	—	—	—

[5]With an exception for `MS MeshGraphNet`, which only supports a batch size of $B = 1$, resulting in $500\,\mathrm{k}$ weight update steps.

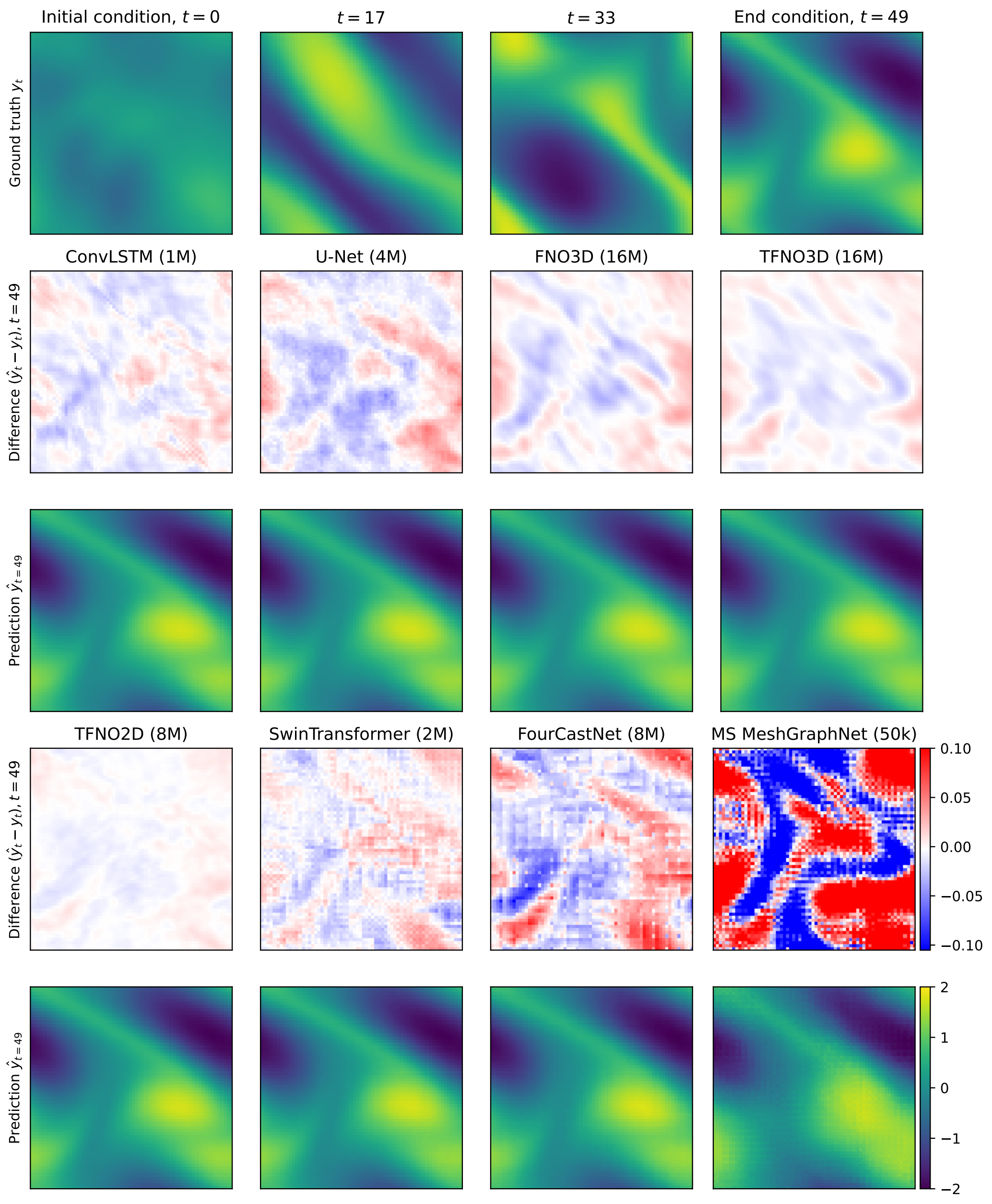

Figure 2: Qualitative results on the Navier-Stokes dataset with Reynolds Number $Re = 1 \times 10^3$ trained on $1\,\mathrm{k}$ samples (experiment 1). The first row shows the ground truth at four different points in time. The remaining rows show the difference between the predicted- and ground-truth at final time (row two and four), as well as the predicted final frame (row three and five). All models receive the first 10 frames of the sequence to predict the remaining 40 frames. The last frame of the predicted sequence from the best models are visualized and respective parameter counts are displayed in parenthesis.

lowest error scores reported in Table 4, we only visualize the best performing model among all parameter ranges in Figure 2 and observe the trend that `TFNO2D` performs best, followed by `TFNO3D`, `SwinTransformer`, `FNO3D`, `ConvLSTM`, `U-Net`, `FourCastNet`, and `MS MeshGraphNet`.

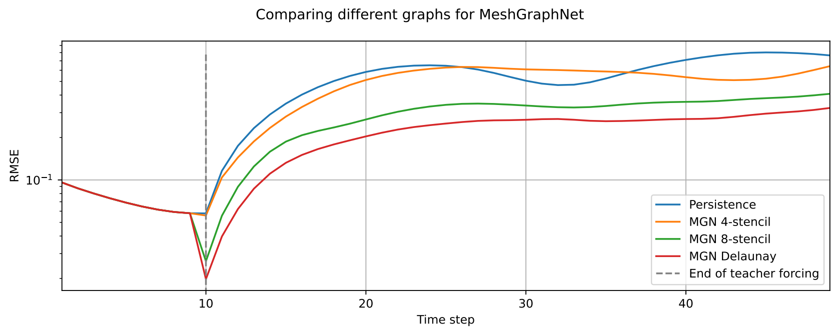

Figure 3: RMSE evolving over forecast time for three different underlying graphs (meshes) that are used in the single scale MeshGraphNet (MGN) (Pfaff et al., 2020).

Next, we study the effect of the underlying graph in GNNs. Observing the poor behavior of `MS MeshGraphNet` in Figure 2, we investigate the effect of three different periodic graph designs to represent the neighborhoods in the GNN. First, the 4-stencil graph connects each node's perpendicular four direct neighbors (i.e., north, east, south, and west) in a standard square Cartesian mesh. Second, the 8-stencil graph adds the direct diagonal neighbors to the 4-stencil graph. Third, the Delaunay graph connects all nodes in the graph by means of triangles, resulting in a hybrid of the 4-stencil and 8-stencil graph, where only some diagonal edges are added. To simplify the problem, we conduct this analysis on the single-scale MeshGraphNet (Pfaff et al., 2020) instead of using the hierarchical `MS MeshGraphNet` (Fortunato et al., 2022). While the graphs have the same number of nodes $|\mathcal{N}| = 4096$, their edge counts differ to $|\mathcal{E}_4| = 16384$, $|\mathcal{E}_8| = 32768$, and $|\mathcal{E}_D| = 24576$ for the 4-stencil, 8-stencil, and Delaunay graph, respectively. The results reported in this paper are based on the 4-stencil graph.

Interestingly, as indicated in Figure 3, the results favor the Delaunay graph over the 8- and 4-stencil graphs, respectively. Apparently, the increased connectedness is beneficial for the task. At the same time, though, the irregularity introduced by the Delaunay triangulation potentially forces the model to develop more informative codes for the edges to represent direction and distance of neighbors more meaningfully.

Lastly, Figure 4 compares the RMSE, memory consumption and computational cost in seconds per epoch as a function of the number of parameters. We see that `TFNO2D L4` performs the best in terms of the RMSE and also scales well with respect to memory and runtime.

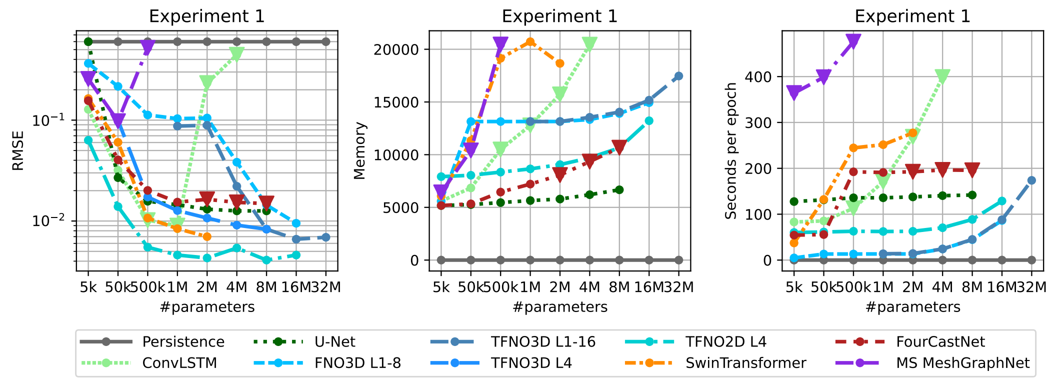

Figure 4: RMSE (left), memory consumption (center), and runtime complexity in seconds per epoch (right) over different parameter counts for models trained on Reynolds Number $Re = 1 \times 10^3$ with 1 k samples for experiment 1.

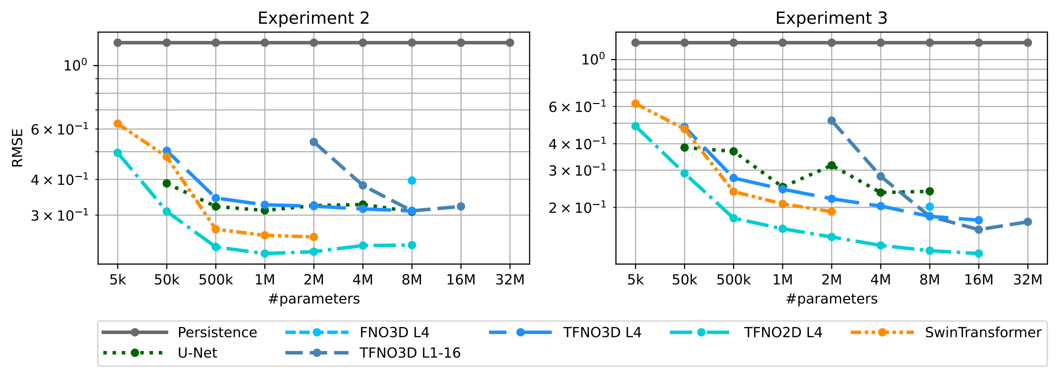

Figure 5: RMSE vs. parameters for models trained on Reynolds Number $Re = 1 \times 10^4$ with $1\,\mathrm{k}$ (experiment 2, left) and $10\,\mathrm{k}$ (experiment 3, right) samples. Note the different y-axis scales. Main observation: As expected, model performance correlates with the number of samples. The number of samples, though, does not affect the model ranking.

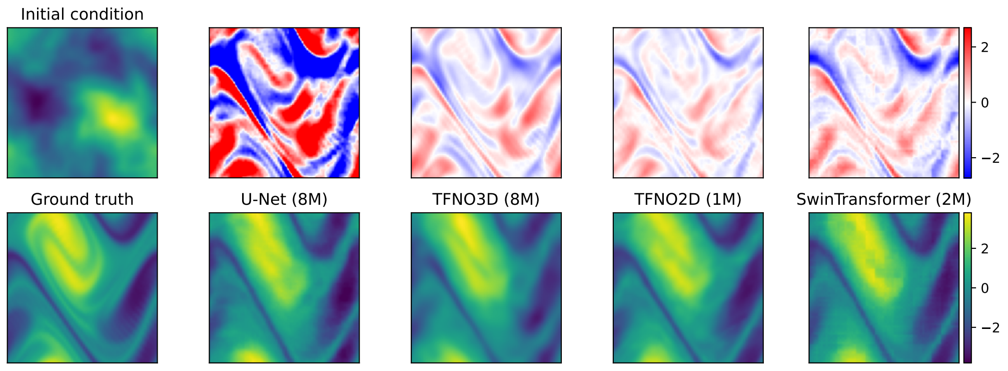

Figure 6: Qualitative results on Navier-Stokes data with Reynolds Number 1×10^4 trained on $1\,\mathrm{k}$ samples (experiment 2). The top left shows the initial condition. The remaining columns in the top row show the differences between the predicted and ground-truth at the final time for the various models. The bottom left shows the ground truth at the final time. The remaining columns in the bottom row show the final predictions from the various models to visually compare to the ground truth. All models face difficulties at resolving the yellow vortex, resulting in blurry predictions around the turbulent structure at this higher Reynolds Number. Among the parameter ranges, the best models are selected for visualizations (parameter count in brackets).

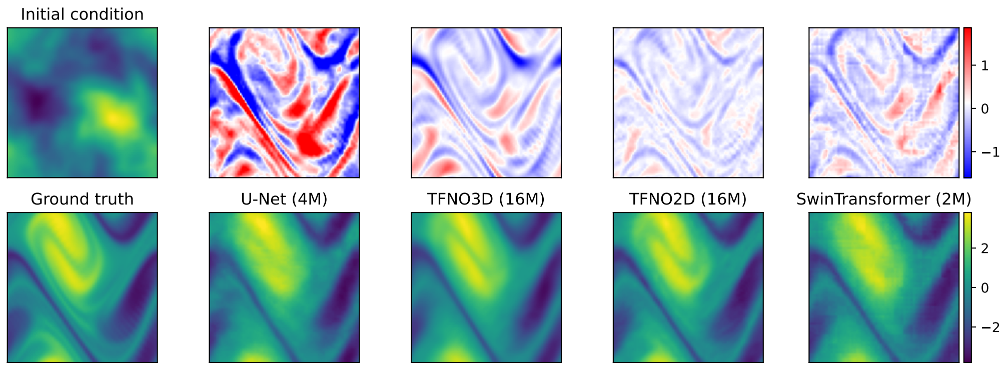

Figure 7: Qualitative results on Navier-Stokes data with Reynolds Number 1×10^4 trained on $10\,\mathrm{k}$ samples (experiment 3). In comparison to Figure 6, the yellow vortex is captured more accurately by `TFNO2D` as a consequence of the larger training set. See plot description in Figure 6 for details.

B.2 Results from Experiment 2 and Experiment 3: $Re = 1 \times 10^4$

Table 4 shows the quantitative error scores of all the experiments (for an easier comparability). We see that the same trend occurs across all three experiments with `TFNO2D` performing the best. Figure 5 illustrates the similar trends of these RMSE results from experiments 2 and 3. Figure 6 and Figure 7 provide the qualitative visualizations for experiments 2 and 3, respectively. Figure 5 (right) and Figure 7 for experiment 3 show that, while all models consistently improve their scores due to the larger training set, the results from experiments 1-2 still hold. That is, when comparing the convergence levels in Figure 5 (right) and Table 4, we see that all models saturate at lower error regimes, while the ordering of the model performance from experiment 1 remains unchanged. Also qualitatively, the models benefit from the increase of training samples in experiment 3, since, visually, the yellow vortex at this higher Reynolds Number is resolved more accurately when the models are trained on more data, as illustrated in Figure 6 and Figure 7.