Lightweight Fourier Neural Operator for Time-Dependent Partial Differential Equations

Dawon Ahn
Department of Computer Science & Engineering
University of California, Riverside
dahn017@ucr.edu

Satish Chandran
Department of Mathematics
University of California, Riverside
schan360@ucr.edu

Daniel Leibovici
NVIDIA
dleibovici@nvidia.com

Nikola Kovachki
NVIDIA
nkovachki@nvidia.com

Evangelos E. Papalexakis
Department of Computer Science & Engineering
University of California, Riverside
epaplex@cs.ucr.edu

Jean Kossaifi
NVIDIA
jkossaifi@nvidia.com

Abstract

Fourier Neural Operators (FNOs) have shown strong performance in solving time-dependent partial differential equations (PDEs). However, accurately modeling complex spatio-temporal dynamics remains challenging and is typically addressed in one of two ways: (i) by applying spectral convolutions over the spatial domain with temporal dynamics handled autoregressively, or (ii) by applying spectral convolutions over the entire spatio-temporal domain. While the former is more computationally efficient, it fails to capture true spatio-temporal interactions. The latter, though more accurate, becomes computationally prohibitive when scaling to larger datasets. We propose LITEFNO, a novel FNO framework that achieves both numerical accuracy and computational efficiency for time-dependent PDEs. Specifically, we first model spatial dynamics by learning a low-rank spatial basis of spectral convolutional weights space. We then incorporate temporal dynamics by learning a new temporal basis through transduction. This factorized formulation enables efficient learning of full spatio-temporal dynamics with significantly fewer parameters (99.9% reduction) and superior performance (44% improvement in VRMSE) compared to the variants of FNO models. The source code and the dataset are available at `https://anonymous.4open.science/r/LFNO`.

1 Introduction

Time-dependent partial differential equations (PDEs) are widely used to model physical systems that evolve over space and time such as fluid dynamics, thermodynamics, and wave propagation [1, 2, 3, 4, 5]. Time-*dependent* PDEs generally exhibit complex behavior as their solutions exist in a higher-dimensional manifold space, in contrast to time-*independent* PDEs whose solutions only vary in space. Traditional numerical methods (e.g., finite volume methods [6, 7], finite element methods [8, 9], etc.) have been used to solve these PDEs [10], but their high computational cost can be a bottleneck for large-scale applications such as weather forecasting [11]. To address these challenges, machine learning–based PDE surrogate models have been introduced, offering significantly faster inference once trained compared to traditional solvers.

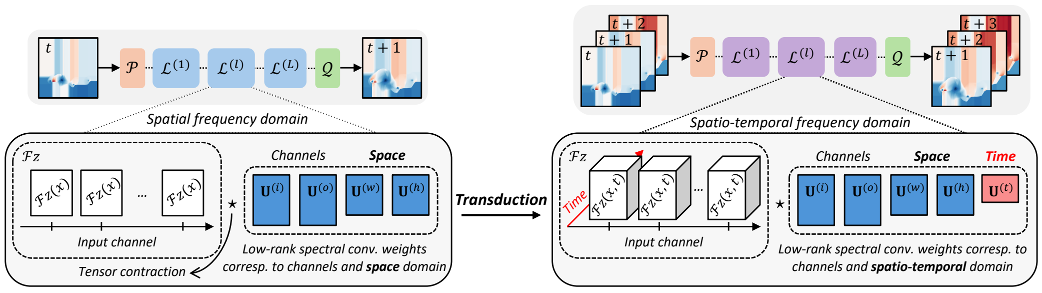


Figure 1: **Overview of the proposed framework**. LITEFNO efficiently learns temporal dynamics based on spatial dynamics through transduction. We model spatial dynamics by learning low rank spatial convolution weights in Fourier space by predicting the next single frame. We extend this model to learn temporal dynamics by predicting the temporal sequences, where each timestep is shifted by one. This transduction requires only one additional low-rank factor along the time domain, enabling spatio-temporal convolution in Fourier space.

Fourier neural operators (FNOs) [12] have emerged as a powerful class of PDE surrogate models which learn operators that map between function spaces [13]. FNOs can generalize across different physical/input parameters within the same PDE family, whereas other surrogate models often require retraining for each new parameter setting [14, 15, 16]. By learning PDE solutions in the Fourier space, FNOs achieve resolution-invariance and are capable of zero-shot super-resolution. This allows FNOs trained on coarse grids to infer accurate solutions at higher grid resolutions.

Despite their strong performance, FNOs still struggle to learn complex spatio-temporal dynamics. One remedy consists of performing the spectral convolutions in the spatial frequency domain to learn the spatial dynamics while predicting the temporal dynamics autoregressively [12, 17, 18]. This approach is computationally efficient, but it struggles to learn complex spatio-temporal dynamics as it does not account for the higher frequencies in the temporal dynamics, i.e. it assumes the solution space is not rapidly changing in time. However, this assumption does not hold for unstable and turbulent systems. Another solution consists of performing the spectral convolutions over the entire spatio-temporal frequency domain, allowing the model to learn both dynamics simultaneously. However, this approach incurs significantly higher computational costs, making it infeasible for large-scale systems. The main computational bottleneck comes from the dense spectral convolution weights, whose parameter count grows exponentially with domain size. Thus, reducing the computational cost of spectral convolution weights is the key factor to efficiently learn spatio-temporal dynamics in Fourier space. To address this issue, we propose LITEFNO, a FNO framework that accurately and efficiently learns spatio-temporal dynamics for time-dependent PDEs.

2 Methodology

In this section, we review FNOs and explain our novel method LITEFNO as illustrated in Figure 1.

Operator Learning. Let $\mathcal{A} := \{a : D_{\mathcal{A}} \to \mathbb{R}^{d_{\mathcal{A}}}\}$ and $\mathcal{U} := \{u : D_{\mathcal{U}} \to \mathbb{R}^{d_{\mathcal{U}}}\}$ denote two input and output function spaces, where $D_{\mathcal{A}}, D_{\mathcal{U}}$ are subsets of a Banach space $D \in \mathbb{R}^d$. An operator $\mathcal{G} : \mathcal{A} \to \mathcal{U}$ is a mapping between the two function spaces. A neural operator aims to approximate an operator $\mathcal{G}$ by constructing a parameter map $\mathcal{G}^\dagger : \mathcal{A} \times \Theta \to \mathcal{U}$ (or equivalently $\mathcal{G}_\theta : \mathcal{A} \to \mathcal{U}$), trained from N input-solution pairs $\{(a_j, u_j)\}_{j=1}^N$ where $u_j = \mathcal{G}(a_j)$.

Fourier Neural Operator. Li et al. [12] formulate FNOs as an iterative architecture to map an input function a to an output function u as follows.

$$u = \mathcal{G}^\dagger(a) = (\mathcal{Q} \circ \mathcal{L}^{(1)} \circ \cdots \circ \mathcal{L}^{(L)} \circ \mathcal{P})(a), \tag{1}$$

where $\circ$ indicates a composition of layers. $\mathcal{Q}, \mathcal{P}$, and $\mathcal{L}^{(l)} (1 \le l \le L)$ indicate projection, lifting, and the l-th Fourier operator layer, respectively. The l-th operator layer maps its hidden representation to the next layer with $\mathcal{L}^{(l)} : \mathbb{R}^{d_{z_l}} \to \mathbb{R}^{d_{z_{l+1}}}$. The input function a is lifted to a higher dimensional representation by the local transformation $\mathcal{P} : \mathbb{R}^{d_{\mathcal{A}}} \to \mathbb{R}^{d_{z_0}}$. The last hidden representation z_L of the L-th layer maps to the final output by the local transformation $\mathcal{Q} : \mathbb{R}^{d_{z_L}} \to \mathbb{R}^{d_{\mathcal{U}}}$.

Omitting the layer index l for simplicity, each Fourier operator layer is defined as:

$$\mathcal{L}(z) = \sigma\big(Wz + b + \mathcal{K}(z)\big), \tag{2}$$

where σ is a non-linear activation function, W is the weight, and b is the bias term. $\mathcal{K}$ indicates a Fourier integral operator which performs spectral convolution in the Fourier space, i.e.

$$\mathcal{K}(z) = \mathcal{F}^{-1}\big(\mathcal{F}(\kappa) \cdot \mathcal{F}(z)\big) = \mathcal{F}^{-1}\big(\boldsymbol{\mathcal{W}} \cdot \mathcal{F}(z)\big) \;\; \forall z \in \mathbb{R}^{s_1 \times \cdots s_d \times m}, \tag{3}$$

where s_d denotes the resolution of the d-th domain, $\mathcal{F}$ and $\mathcal{F}^{-1}$ indicate the Fourier transform and its inverse respectively, and $(\cdot)$ indicates a matrix multiplication along the last dimension. Note that κ is a kernel function and is replaced to a parameter tensor $\boldsymbol{\mathcal{W}} \in \mathbb{C}^{k_1 \times \cdots \times k_d \times m \times n}$ to be learned directly in Fourier space where k_d indicates the truncated Fourier modes for the d-th domain and m, n indicates the input and output channel size. Here, $\boldsymbol{\mathcal{W}}$ is the main bottleneck for the scalability and computation, with a complexity of $O(mnk^d)$, where $k = max(k_1, \cdots, k_d)$. The size of the weight exponentially increases with respect to d, leading to a significant increase in computations particularly for convolving across the spatio-temporal domain.

Low-Rank Factorized Spectral Convolution. We replace a dense weight $\boldsymbol{\mathcal{W}}$ with compact R low-rank factors using CP decomposition [19, 20] as follows.

$$\boldsymbol{\mathcal{W}} = \sum_{r=1}^{R} \lambda_r \mathbf{u}_r^{(1)} \circ \cdots \circ \mathbf{u}_r^{(d)} \circ \mathbf{u}_r^{(i)} \circ \mathbf{u}_r^{(o)} = [\boldsymbol{\lambda}; \mathbf{U}^{(1)}, \cdots, \mathbf{U}^{(d)}, \mathbf{U}^{(i)}, \mathbf{U}^{(o)}], \tag{4}$$

where $\circ$ denotes an outer product, $\boldsymbol{\lambda} \in \mathbb{C}^R$ is a weight vector for rank-one tensors. Note that $\mathbf{U}^{(j)}$ $(j \in \{1, \ldots, d, i, o\})$ are low-rank bases representing input, output, and domain modes of spectral convolution weights. Expressing the convolution weights $\boldsymbol{\mathcal{W}}$ into low-rank factorized form has three advantages: model parameter reduction, computational efficiency, and regularization effects [21]. Leveraging the low-rank form reduces the complexity from $O(mnk^d)$ to $O((m + n + kd)R)$, where $R << m, n, k$. We can directly contract the input with those low-rank factors, rather than reconstructing a full tensor weight and multiplying them to the input $\mathcal{F}z$.

Transduction: From Spatial to Spatio-Temporal Learning. We introduce a transduction framework to extend spatial learning to spatio-temporal learning, motivated by prior work in computer vision [22]. The model first learns a low rank basis of spatial dynamics in the Fourier domain with Equation (4), where it is trained to predict the next timestep at $t + 1$ given input time step t. We expand single timestep input to segments of timesteps and apply the Fourier transform to the entire spatio-temporal domain to learn spatio-temporal dynamics in the Fourier domain. This expanded input requires an additional low-rank factor $\mathbf{U}^{(t)} \in \mathbb{C}^{k_t \times R}$ corresponding to time domain in Equation (4) in the Fourier layer. To learn spatio-temporal dynamics while preserving the modeled spatial dynamics, we fine-tune the expanded model over temporal windows from $t : t+s$ to $t+1 : t+s+1$. This allows the model to avoid retraining from scratch, thereby significantly reducing the training time.

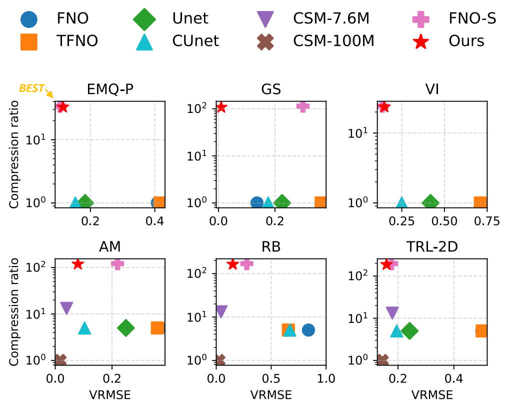
3 Experiments

All experiments are conducted using a single GPU machine with a NVIDIA RTX A6000 or A4000.

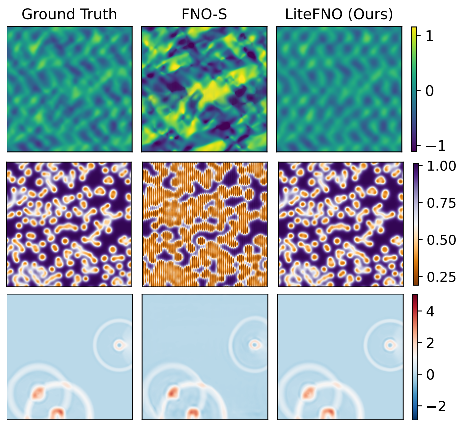
Dataset. We select a series of diverse benchmark datasets from *"The Well"*[1] [23], which consists of many chaotic and turbulent spatio-temporal physics simulations. We use the following eight datasets: Active matter (AM), Euler multi-quadrants with open boundary conditions (EMQ-O), Euler multi-quadrants with periodic boundary conditions (EMQ-P), Acoustic scattering single discontinuity (AS-SD), Gray Scott reaction diffusion (GS), Rayleigh Benard (RB), Turbulent Radiative Layer-2D (TRL-2D), and Viscoelastic Instability (VI). Each dataset is split into training, validation, and test dataset. Due to computational constraints, we reduce the training data by: 1) randomly sampling 1000 trajectories for datasets with more than 1000 trajectories, 2) using only the first 60 timesteps, and 3) downsampling spatial resolutions by a factor of 4 or 8. Details on original and training datasets are provided in Appendix A.1.

Model and training. We have three baselines, FNO-S, FNO-ST, and FNO-ST (FULL), sharing the same FNO architecture from [21], which shows the state-of-the-art performance among

[1] `https://polymathic-ai.org/the_well/`



(a) Compression ratio v.s. VRMSE



(b) Visualization of prediction

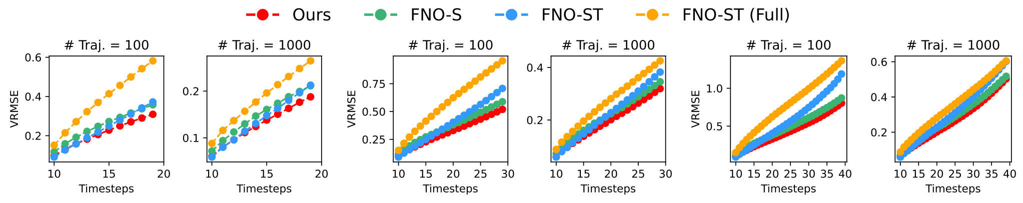
Figure 2: **Overall performance comparison of our methods compared with the prior state of the art methods**. Figure 2(a) shows that LITEFNO achieves the best trade-off between compression ratio and VRMSE in learning complex time-dependent PDEs datasets. Figure 2(b) visualizes the spatial super-resolution of autoregressive prediction of FNO-S and LITEFNO on AM, GS, and AS-SD datasets.

FNO models. FNO-S and FNO-ST use low rank spectral convolution weights for the spatial and spatial-temporal domains, respectively, while FNO-ST (FULL) uses full dense spectral convolution weights along the spatio-temporal domain. All models have eight layers, and their width is selected from $\{64, 128, 160\}$. The number of Fourier modes is set to be less than two-thirds of the input resolution, and the rank is chosen from $\{32, 48\}$. We train models for 500 epochs using an AdamW optimizer [24] with an initial learning rate of 0.001, halved every 100 epochs. For transduction, we fine-tune the proposed model for 100 epochs, where the learning rate is halved every 20 epochs. We select the weight decay factor from $\{10^{-5}, 5 \times 10^{-3}\}$, and the batch size of 16 or 32. For the training losses, we use the sum of the relative L^2 norm over the spatial domain for FNO-S and the spatio-temporal domain for FNO-ST and LITEFNO. The time window S for input and output sequences is set to 3 for models convolving in the spatio-temporal domain. Full model configurations and training details are provided in Appendices A.2 and A.3.

Metrics. We evaluate baselines with Variance Scaled Root Mean Squared Error (VRMSE) [23], defined as $\mathrm{VRMSE}(u, v) = \left(\langle |u - v|^2\rangle / (\langle |u - \bar{u}|^2\rangle + \epsilon)\right)^{1/2}$, where $\langle \cdot \rangle$ denotes the spatial mean operator and the $\epsilon = 10^{-7}$ term prevents division by zero in cases where $\langle |u - \bar{u}|^2 \rangle$ is zero.

Result1: Comparison Against Benchmarks. Figure 2 compares the performance of benchmark models and LITEFNO under the VRMSE for one-step prediction task. We provide the full results of multi-step and one-step prediction tasks in Appendix B.1 with tasks and benchmarks explanation in Appendices A.4 and A.5. There are two key points to highlight here. First, all benchmark models are trained on full datasets (i.e. the entire trajectories, time sequences, and at original resolutions) while our model is trained on reduced datasets. As such, FNO-S and LITEFNO produce super-resolution outputs over the spatial domain. Despite this, our model achieves lower error with significantly fewer parameters and less training data as shown in Fig. 2(a). Second, LITEFNO achieves better performance than FNO-S with only a slight increase in model size thanks to the low rank spectral convolution weights. Even though they share the same architecture, the error gap comes from the fact that FNO-S learns spatial dynamics while LITEFNO learns spatio-temporal dynamics in Fourier space. As shown in Fig. 2(b), the autoregressive predictions of LITEFNO captures complex dynamics while FNO-S produce aliasing (AM), inaccurate (GS) or distorted predictions (AS-SD).

Result 2: Comparison Against FNO Variants. We evaluate the predictive performance of LITEFNO with FNO variants under a set of diverse physical scenarios. The prediction time ranges vary from $10:20$ to $10:40$, while the input time range is fixed from $0:10$. To evaluate the data efficiency of the model, we vary the time sequences used in training from 20 to 50, and evaluate performance in both the low-data regime (100 trajectories) and the high-data regime (1000 trajectories). Figure 3 shows the VRMSE of all models on EMQ-O dataset for each different prediction time range. Overall, LiteFNO consistently achieves the lowest VRMSE compared to all baselines in all prediction tasks for the entire timesteps.



(a) 10 : 20 (b) 10 : 30 (c) 10 : 40

Figure 3: **Comparison of the predictive performance** of our method compared with FNO variants, over the time range 10 to 40, while varying the number of trajectories on the EMQ-O dataset.

4 Conclusion

We propose LiteFNO, an accurate and efficient Fourier neural operator architecture for time-dependent PDEs. The model achieves superior accuracy and computational efficiency by replacing full dense spectral convolution weights with learned low rank bases of the weights, extended to the temporal domain through transduction. Our results demonstrate that LiteFNO creates a pathway for an accurate and efficient surrogate modeling for complex spatio-temporal PDE systems.

Acknowledgments and Disclosure of Funding

This research was supported in part by the National Science Foundation under CAREER grant no. IIS 2046086 and CREST Center for Multidisciplinary Research Excellence in CyberPhysical Infrastructure Systems (MECIS) grant no. 2112650, and by the Agriculture and Food Research Initiative Competitive Grant no. 2020-69012- 31914 from the USDA National Institute of Food and Agriculture. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

References

[1] Lawrence C. Evans. *Partial differential equations*. American Mathematical Society, Providence, R.I., 2010.

[2] Clive L. Dym and Irving H. Shames. *Solid Mechanics: A Variational Approach, Augmented Edition*. Springer New York, New York, NY, 2013.

[3] Ronald L. Panton. *Incompressible Flow*. John Wiley & Sons, Ltd, 2013.

[4] J. D. Jackson. *Electrodynamics, Classical*. John Wiley & Sons, Ltd, 2003.

[5] R Byron Bird, Warren E Stewart, and Edwin N Lightfoot. *Transport Phenomena*. John Wiley & Sons, Chichester, England, November 2006.

[6] Randall J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations*. Society for Industrial and Applied Mathematics, 2007.

[7] Randall J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2002.

[8] Fredrik Bengzon Mats G. Larson. *The Finite Element Method: Theory, Implementation, and Applications*. Springer Berlin, Heidelberg, 2013.

[9] L. Ridgway Scott Susanne C. Brenner. *The Mathematical Theory of Finite Element Methods*. Springer New York, NY, 2008.

[10] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, USA, 3 edition, 2007.

[11] Wenhui Peng, Shaoxiang Qin, Senwen Yang, Jianchun Wang, Xue Liu, and Liangzhu (Leon) Wang. Fourier neural operator for real-time simulation of 3d dynamic urban microclimate. *Building and Environment*, 248:111063, 2024.

[12] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[13] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

[14] Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 481–490, 2016.

[15] Yinhao Zhu and Nicholas Zabaras. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415–447, 2018.

[16] Yuehaw Khoo, Jianfeng Lu, and Lexing Ying. Solving parametric pde problems with artificial neural networks. *European Journal of Applied Mathematics*, 32(3):421–435, 2021.

[17] Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized fourier neural operators. *arXiv preprint arXiv:2111.13802*, 2021.

[18] Michael McCabe, Peter Harrington, Shashank Subramanian, and Jed Brown. Towards stability of autoregressive neural operators. *arXiv preprint arXiv:2306.10619*, 2023.

[19] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[20] Nicholas D. Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E. Papalexakis, and Christos Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582, 2017.

[21] Jean Kossaifi, Nikola Kovachki, Kamyar Azizzadenesheli, and Anima Anandkumar. Multi-grid tensorized fourier neural operator for high-resolution pdes. *arXiv preprint arXiv:2310.00120*, 2023.

[22] Jean Kossaifi, Antoine Toisoul, Adrian Bulat, Yannis Panagakis, Timothy M Hospedales, and Maja Pantic. Factorized higher-order cnns with an application to spatio-temporal emotion estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6060–6069, 2020.

[23] Ruben Ohana, Michael McCabe, Lucas Meyer, Rudy Morel, Fruzsina Agocs, Miguel Beneitez, Marsha Berger, Blakesly Burkhart, Stuart Dalziel, Drummond Fielding, et al. The well: a large-scale collection of diverse physics simulations for machine learning. *Advances in Neural Information Processing Systems*, 37:44989–45037, 2024.

[24] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[25] Payel Mukhopadhyay, Michael McCabe, Ruben Ohana, and Miles Cranmer. Compute-adaptive surrogate modeling of partial differential equations. In *ICLR 2025 Workshop on Machine Learning Multiscale Processes*.

[26] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.

[27] 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. *arXiv preprint arXiv:2010.11929*, 2020.

A Experimental Setting

We provide additional details on the dataset in Appendix A.1, model architecture and its size in Appendix A.2, training in Appendix A.3, evaluation in Appendix A.4, and benchmark in Appendix A.5 to further clarify the experimental setups.

A.1 Dataset

We summarize datasets used for training models in Appendix A.1.1.

A.1.1 Summary

Tables 1 and 2 summarize the selected eight original benchmark datasets and the corresponding modified training datasets. Each trajectory includes the 2D spatial domain ($W \times H$) that evolve over time (T), along with physics fields (F). Accordingly, the input sample to FNO-S has the shape of $F \times W \times H$ while the input sample to LITEFNO, FNO-ST, and FNO-ST (FULL) has the shape of $F \times W \times H \times S$ where S is the time window, as summarized in Table 3. To reduce a size of dataset, we randomly select 1000 trajectories from the EMQ-O, EMQ-P, AS-SD, and RB datasets. Also, we select initial 30-60 time steps for all datasets except for AM, RB, VI since these three datasets do not have enough trajectories to train with. For spatio-temporal super-resolution tasks, we use the entire time steps for EMQ-O, AS-AS, GS since the temporal resolution is downsampled by a factor of 4, thus reducing the time steps from 100 to 25.

Table 1: Summary of original datasets.

Dataset	Name	Train/val/test	$W \times H \times T$	# Fields
`euler_multi_quadrants_openBC`	EMQ-O	4000/500/500	$512 \times 512 \times 100$	5
`euler_multi_quadrants_periodicBC`	EMQ-P	4000/500/500	$512 \times 512 \times 100$	5
`acoustic_scattering_discontinuous`	AS-SD	1600/200/200	$256 \times 256 \times 100$	3
`active_matter`	AM	175/24/26	$256 \times 256 \times 80$	11
`gray_scott_reaction_diffusion`	GS	960/120/120	$128 \times 128 \times 100$	2
`rayleigh_benard`	RB	1400/175/175	$512 \times 128 \times 100$	4
`turbulent_radiative_layer_2D`	TRL-2D	72/9/9	$128 \times 384 \times 100$	4
`viscoelastic_instability`	VI	213/22/22	$512 \times 512 \times 19$	8

Table 2: Summary of training datasets.

Dataset	Name	Train/val/test	$W \times H \times T$	# Fields
`euler_multi_quadrants_openBC`	EMQ-O	1000/500/500	$64 \times 64 \times 40$	5
`euler_multi_quadrants_periodicBC`	EMQ-P	1000/500/500	$64 \times 64 \times 40$	5
`acoustic_scattering_discontinuous`	AS-SD	1000/200/200	$64 \times 64 \times 60$	3
`active_matter`	AM	175/24/26	$64 \times 64 \times 80$	11
`gray_scott_reaction_diffusion`	GS	960/120/120	$64 \times 64 \times 60$	2
`rayleigh_benard`	RB	1000/175/175	$64 \times 16 \times 60$	4
`turbulent_radiative_layer_2D`	TRL-2D	72/9/9	$32 \times 96 \times 100$	4
`viscoelastic_instability`	VI	213/22/22	$64 \times 64 \times 19$	8

Table 3: A shape of input and output training samples depending on the domain which FNO handles. Here B, C, W, H, and S denotes number of batch size, channel, width, height, and time windows respectively. Note that C equals to F and t $(1 \leq t \leq T)$ is an index of time steps.

Domain	**Input**	**Output**
Space	$u(\cdot, t) \in \mathbb{R}^{B \times C \times W \times H}$	$u(\cdot, t+1) \in \mathbb{R}^{B \times C \times W \times H}$
Space and time	$\{u(\cdot, t)\}_{t}^{t+S} \in \mathbb{R}^{B \times C \times W \times H \times S}$	$\{u(\cdot, t)\}_{t+1}^{t+S+1} \in \mathbb{R}^{B \times C \times W \times H \times S}$

A.2 Model

We chose the improved FNO architecture by [21]. In the lifting and projection layers, two point-wise MLPs are applied to the channel dimensions. As illustrated in Figure 4, each Fourier layer consists of a low-rank factorized spectral convolution, a channel-wise MLP, and two skip connections to enhance the model capacity. The first skip connection applies 1D convolution to flattened all domain dimensions, and the second skip connection uses soft-gating which re-weights the channel dimension. Also, we compare the size of the spectral convolution weights in Table 4, showing that our approach significantly reduces the number of parameters due to the low rank factorization. We also report the exact configuration and model size of models used in Table 5. For Figure 3, the model size for FNO-ST (FULL) is 805,643,100, while the size of LITEFNO is 502,152, reducing the model size by 99%.

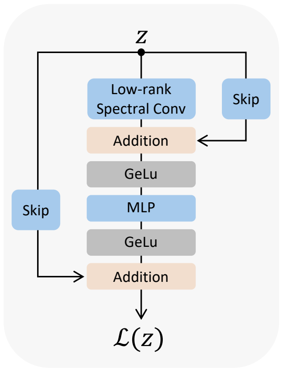


Figure 4: Architecture of Fourier layer

Table 4: Comparison of parameter size of spectral convolution in all Fourier layers between the original FNO and its variants. Here, m and n denote channel sizes respectively, and k is the maximum Fourier modes, d is the number of domain dimensions, and l is the number of layers. In typical setting, $m, n >> l, d, k$ such that LITEFNO significantly reduces the memory footprint by applying the low-rank factorization to the spectral weights.

Model	Number of Parameters of Spectral Convolution in All Layers
Original [12]	$O(mnlk^d)$
FFNO [17]	$O(mnkd)$
LITEFNO [Ours]	$O((m+n+k)ld)$

Table 5: Model configurations and size. The layer is set to 8 and time window S is set to 3 for LITEFNO. Note that FNO-ST has the same parameter size as LITEFNO.

Dataset	Model Width	Rank	Fourier Modes		Model Size	
			FNO-S	LITEFNO	FNO-S	LITEFNO
EMQ-O	128	32	32,16	32,32,2	492,680	502,152
EMQ-P	128	48	48,24	48,48,2	589,192	609,416
AS-SD	64	32	32,16	32,32,2	176,968	186,824
AM	160	48	48,24	48,48,2	824,872	845,160
GS	64	32	32,16	32,32,2	176,968	186,824
RB	128	48	48,8	48,16,2	576,904	584,840
TRL-2D	128	32	32,48	32,96,2	509,064	534,920
VI	160	48	48,24	48,48,2	824,872	845,160

A.3 Training

We explain the detailed initialization procedure used in transduction from FNO-S to FNO-ST. As the input domain expands from spatial to spatio-temporal, additional weights are introduced into the model. These newly introduced weights are carefully initialized to ensure compatibility with the pre-trained weights. In the lifting layer, the FNO takes an input which concatenates grid embeddings corresponding to the domain dimensions to the input sample, along the channel dimensions. For example, a 2D spatial domain requires 2-dimensional grid embeddings, while a 2D spatial + 1D temporal domain requires 3-dimensional grid embeddings. When the model is extended from the spatial domain to the spatio-temporal domain, the the dimension of grid embeddings increases by one. As a result, the first dimension of the weight matrix of the first MLP in the lifting layer is expanded by one to handle the extended input. We initialize the corresponding weights to zero such that they do not interfere with the pre-trained weights of the lifting layer. In the Fourier layer, we introduce new low-rank factorized spectral convolution weights corresponding to the temporal domain and the last spatial dimension which corresponds to negative frequencies. The Fourier transform (FT) of real-valued signals has conjugate symmetry where negative frequency components are the complex conjugate of the positive ones. As a result, half of the frequency components are redundant to compute the FT. Leveraging this property, our implementation truncates half of the frequencies along the last dimension of the input domain (i.e., height dimension). That is, the low-rank spectral convolutional weights corresponding to the last dimension are trained using only the positive frequency modes in FNO-S. However, FNO-ST requires full frequencies for the last dimension of the spatial domain. Thus, we need to introduce additional low-rank weights corresponding to the negative frequency modes, in addition to introducing those for the time dimension. By leveraging the property of conjugate symmetry, we initialize the weight corresponding to the negative frequencies by mirroring the pre-trained weights corresponding the positive frequencies. For the temporal dimension, we make it $1 + 0.0j$ to every value to avoid interfering the pre-trained weights (here j is the imaginary number). After initializing all the newly introduced weights, we fine tune the model for 100 epochs to transduct FNO-S to FNO-ST. Thanks to the pre-trained weights, the model converges faster and requires fewer training epochs compared to the training the model from scratch.

A.4 Evaluation

There are two types of evaluation tasks: a single-step prediction and a multi-step prediction task. For the single-step prediction task, the model predicts one future step given a specific input sequence. For multiple-step prediction, the model autoregressively predicts the next step up to 33, given the initial four time steps/frames ranging from 0 to 3 (i.e., $u(0), u(1), u(2),$ and $u(3)$). Autoregressive predictions are evaluated over two time windows 6:12 and 13:30 which correspond to the time intervals from 9 to 15 (i.e., $u(9), \cdots, u(15)$) and from 16 to 33 (i.e., $u(16), \cdots, u(33)$), respectively.

A.5 Benchmark

We explain the model architecture and training strategy of the FNO benchmarks. Both FNO and TFNO perform spectral convolution in the spatial frequency domain. They are trained using four historical time steps to predict the next single step. To handle the historical sequence, these models stack them along channel dimensions (physics fields) (i.e., $(F \times T) \times W \times H$) and learn temporal dynamics in physical domain using neural networks. However, our model learn temporal dynamics explicitly in the Fourier domain.

B Experimental Results

B.1 Comparison against benchmarks

We include benchmark results from [23, 25] along with FNO-S, which shares the same architecture as our model. FNO and TFNO follow the original model architecture proposed in [12] while TFNO incorporates Tucker decomposition to reduce the size of dense spectral convolution weights. Unet and CUnet are CNN-based neural network models convolving in the physical domain, where CUnet adopts the advanced CNN layers proposed in [26]. CSM [25] is a recent Vision Transformer [27] based PDE surrogate model. The number below each model indicates its parameter count.

Table 6: **Comparison of model performance on VRMSE for multi-step prediction** over time windows (6:12) and (13:30). Note that FNO-S and LITEFNO results represent super-resolution outputs over the spatial domain. The best and second-best results per time window are denoted in bold and underlined, respectively, and — indicates N/A.

Dataset	FNO [23] (20M)		TFNO [23] (20M)		Unet [23] (20M)		CUnet [23] (20M)		FNO-S (<0.84M)		**LITEFNO** (<0.85M)	
	6:12	13:30	6:12	13:30	6:12	13:30	6:12	13:30	6:12	13:30	6:12	13:30
AS-SD	—	—	—	—	—	—	—	—	0.57	1.11	**0.11**	**0.29**
EMQ-P	1.13	1.37	1.23	1.52	1.02	1.63	4.98	>10	0.22	**0.33**	**0.21**	0.33
GS	0.89	>10	1.54	>10	0.57	>10	0.29	7.62	0.97	1.58	**0.03**	**0.08**
VI	4.11	—	0.93	—	0.89	—	0.52	—	0.50	—	**0.50**	—
AM	>10	>10	7.52	4.72	2.53	2.62	2.11	2.71	1.98	4.38	**0.27**	**0.76**
RB	>10	>10	>10	>10	>10	>10	>10	>10	0.23	1.03	**0.11**	**0.37**
TRL-2D	1.79	3.54	6.01	>10	0.66	1.04	0.54	1.01	**0.42**	**0.85**	0.50	0.92

Table 7: **Comparison of model performance on VRMSE for one-step prediction**. Note that FNO-S and LITEFNO results present super-resolution outputs over the spatial domain. The best and second-best results are denoted in bold and underlined, respectively, and — indicates N/A.

Dataset	FNO [23] (20M)	TFNO [23] (20M)	Unet [23] (20M)	CUnet [23] (20M)	CSM [25] (7.6M)	CSM [25] (100M)	FNO-S (<0.84M)	**LITEFNO** (<0.85M)
AS-SD	—	—	—	—	—	—	0.1397	**0.0494**
EMQ-P	0.4081	0.4163	0.1834	0.1531	—	—	**0.1057**	0.1146
GS	0.1365	0.3633	0.2252	0.1761	—	—	0.299	**0.0098**
VI	0.7212	0.7102	0.4185	0.2499	—	—	**0.1359**	0.1481
AM	0.3691	0.3598	0.2489	0.1034	0.0400	**0.0172**	0.2196	0.0798
RB	0.8395	0.6566	1.4860	0.6699	0.0440	**0.0250**	0.2785	0.1506
TRL-2D	0.5001	0.5016	0.2418	0.1956	0.180	**0.1440**	0.1770	0.1593

B.2 Super-resolution on spatial domain

We visualize a spatial super-resolution of our model compared to FNO-S using GS and AM datasets in Figures 5 and 6.

B.3 Super-resolution on spatio-temporal domain

We compare the spatio-temporal super-resolution performance of FNO variants using VRMSE on the EMQ-O, GS, and AS-SD datasets, as shown in Table 8 and Fig. 8. All models are trained using a temporal resolution of 25 and a spatial resolution of 64. We increase the temporal resolution up to $4\times$, and the spatial resolution by $8\times$, $4\times$, and $2\times$ for each dataset. The evaluation task is to predict time steps ranging from 20 to 40, given input time steps ranging from 0 to 20. Table 8 shows that LITEFNO outperforms FNO-ST in most super-resolution settings across all datasets. For the GS dataset, LITEFNO maintains performance with only a marginal error increase ($\leq$ 0.01) even when the temporal resolution is up to $4\times$ and spatial resolution is up to $2\times$, while FNO-ST shows severe performance degradation. For the AS-SD and EQM-O datasets, both methods show

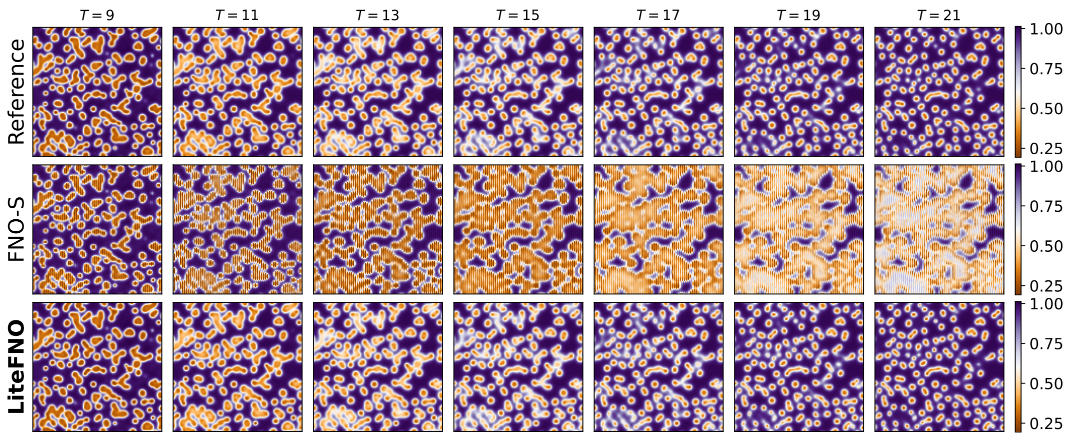


Figure 5: **Super-resolution results over space domain on GS** (128×128).

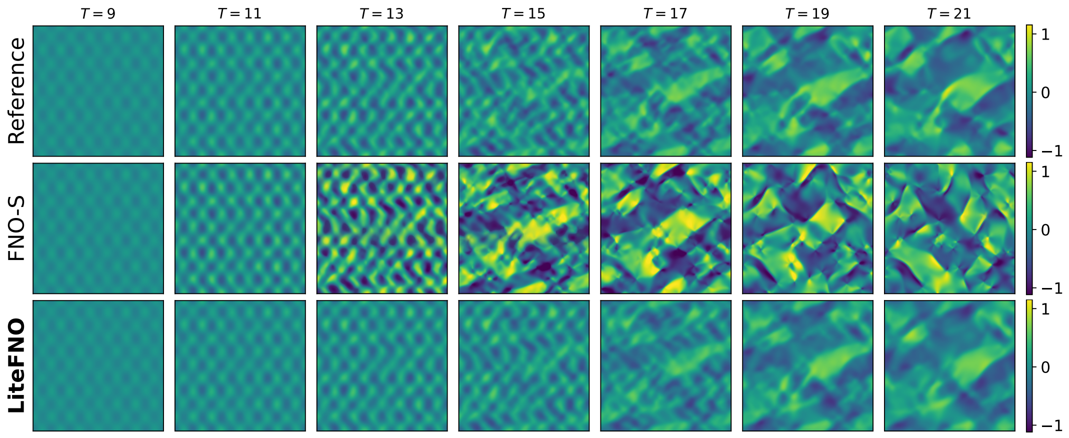


Figure 6: **Super-resolution results over space domain on AM** (256×256).

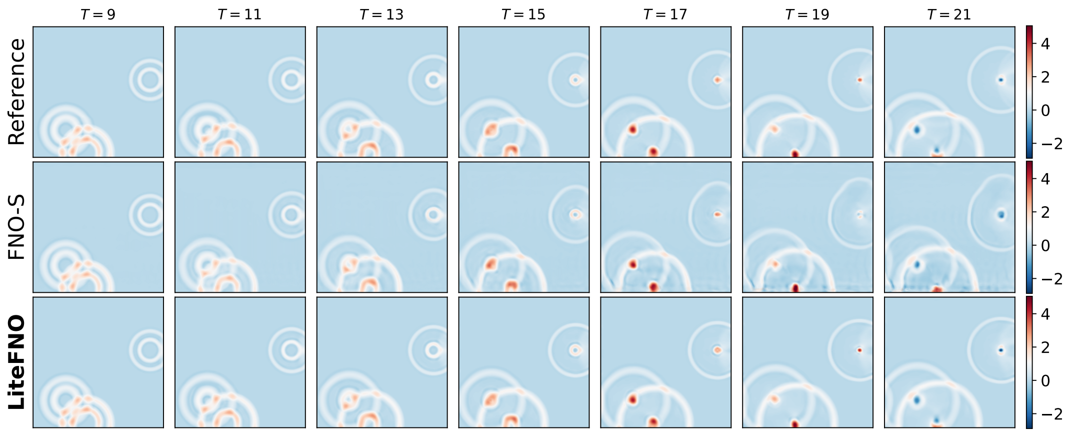
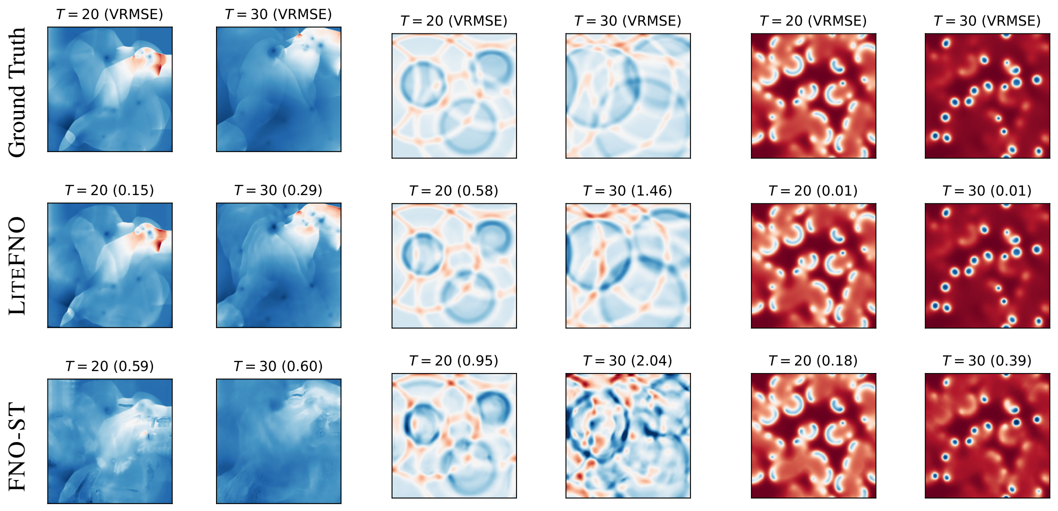


Figure 7: **Super-resolution results over space domain on AS-SD** (256×256).

higher VRMSE under increased temporal resolution compared to spatial resolution, indicating that temporal super-resolution can be more challenging for complex spatio-temporal PDEs. However, LITEFNO shows a relatively smaller increase in VRMSE by 0.36 and 0.75 while FNO-ST shows a larger increase in VRMSE by 0.82 and 1.91 for each dataset. This indicates that LITEFNO performs more robustly in spatio-temporal super-resolution tasks compared to the FNO-ST training full spatio-temporal dynamics from scratch. One key reason for the better super-resolution performance is its transductive framework, which enables the model to learn spatial dynamics accurately. While temporal resolution increases, their temporal dynamics change more slowly, and so spatial dynamics dominate. Thus, accurately learned spatial dynamics help the model to interpolate missing time steps at the finer resolutions. As illustrated in Figure 8(a), LITEFNO captures fine-grained details, while FNO-ST tends to smooth them out.

Table 8: **Comparison of VRMSE in spatio-temporal super-resolution** for multi-step prediction between LITEFNO and FNO-ST on the EQM-O, AS-SD, and GS.

	SR Domain	EQM-O				AS-SD			GS	
Model	Time / Space	1×	2×	4×	8×	1×	2×	4×	1×	2×
LITEFNO	1×	0.15	0.19	0.20	0.21	0.20	0.23	0.25	0.04	0.04
	2×	0.35	0.36	0.37	0.37	0.85	0.84	0.84	0.04	0.04
	4×	0.51	0.51	0.51	0.64	0.57	0.56	0.56	0.03	0.05
FNO-ST	1×	0.15	0.17	0.18	0.18	0.22	0.27	0.29	0.05	0.05
	2×	0.79	0.79	0.79	0.79	1.79	0.94	0.94	20.53	20.54
	4×	0.98	0.98	0.97	0.97	1.16	1.16	1.16	6.62	30.21



(a) EQM-O (b) AS-SD (c) GS

Figure 8: **Super-resolution results over space and time domains** $(128 \times 128 \times 100)$, $(256 \times 256 \times 100)$, and $(512 \times 512 \times 100)$ at timesteps 20 and 40, using EQM-O, AS-SD, and GS, respectively. The rows from top to bottom correspond to a ground truth, LITEFNO, and FNO-S. The numbers in parentheses indicate VRMSE.

B.4 Running time

To evaluate the computational efficiency, we also measure the training time of all baselines while varying the time steps of training dataset, as shown in Figure 9. LITEFNO shows the shortest training time compared to FNO baselines performing spatio-temporal convolution in the Fourier domain. FNO-ST (FULL) takes the longest training time due to its dense spectral convolution weights. By adopting the low-rank spectral convolution weights, FNO-ST reduces the training time by approximately 2×, but it still suffers from the computational burden of the spatio-temporal convolution operations. In contrast, FNO-S shows the shortest training time. Thanks to the transduction framework, LITEFNO avoids training the model from scratch unlike FNO-ST, but is finetuned with fewer epochs.

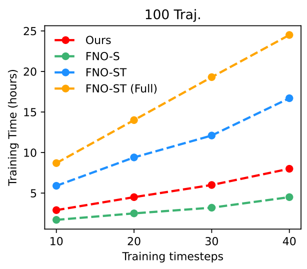


Figure 9: Comparison of training time between baselines. FNO-ST (FULL) shows the longest training time, followed by FNO-ST. FNO-S shows the shortest training time. Although LITEFNO performs the same calculations as FNO-ST, it does not need to train for full epochs from scratch.