Learning the boundary-to-domain mapping using Lifting Product Fourier Neural Operators for partial differential equations

Anonymous Authors[1]

Abstract

Neural operators such as the Fourier Neural Operator (FNO) have been shown to provide resolution-independent deep learning models that can learn mappings between function spaces. For example, an initial condition can be mapped to the solution of a partial differential equation (PDE) at a future time-step using a neural operator. Despite the popularity of neural operators, their use to predict solution functions over a domain given only data over the boundary (such as a spatially varying Dirichlet boundary condition) remains unexplored. In this paper, we refer to such problems as boundary-to-domain problems; they have a wide range of applications in areas such as fluid mechanics, solid mechanics, heat transfer etc. We present a novel FNO-based architecture, named Lifting Product FNO (or LP-FNO) which can map arbitrary boundary functions defined on the lower-dimensional boundary to a solution in the entire domain. Specifically, two FNOs defined on the lower-dimensional boundary are lifted into the higher dimensional domain using our proposed lifting product layer. We demonstrate the efficacy and resolution independence of the proposed LP-FNO for the 2D Poisson equation.

1. Introduction

Computer simulations for real-world science and engineering problems can be complicated to set up, require knowledge of numerical methods, and may take a large amount of computational resources to compute a solution. Furthermore, there are some disciplines where the governing partial differential equations (PDEs) are not known with adequate certainty. In these scenarios, surrogate models are attractive. These may be derived (simplified) from the governing equations by analysis or generated from data gathered in a laboratory or in the real world. In this work, we focus on data-driven surrogate models for phenomena governed by PDEs, particularly neural network-based methods. Neural networks that represent maps from the spatio-temporal domain (the region of interest) to the solution for one instance of a PDE, or family of PDEs parameterized by a few parameters, have been developed over the last several years. This has mainly been in the context of physics-informed neural networks (PINNs) (Raissi et al., 2019; Chen et al., 2020) for both forward and inverse problems. However, PINNs typically do not generalize to an adequately large parameter range (Krishnapriyan et al., 2021). Further the initial and boundary conditions are baked into the model during training and cannot generalize when these are changed.

A different approach is to use an operator approach to map the entire input (a function over the spatial domain or its boundary) to the solution of the PDE, another function over the domain. For example, non-homogeneous material properties $a : \Omega \to \mathbb{R}$ can be mapped to the solution $u : \Omega \to \mathbb{R}$ of the heat equation. If the domain is a rectangle and the input function is discretized on a regular Cartesian grid, this can be done by convolutional neural networks. The 'DeepONet' architecture (Lu et al., 2021) is an operator learning method that uses fully connected or convolutional neural networks to process the input function in the 'branch' sub-network of their architecture, which can then be evaluated at different spatio-temporal coordinates using a separate 'trunk' sub-network. These operator approaches can have a much better payoff for the cost of training the models because the same model can now accept different spatially-varying parameter functions and initial conditions as input and predict the corresponding solution, rather than having them baked-in during training. However, an issue with such models is that they are not fully 'resolution-agnostic'; that is, they are designed for a fixed discretization of the input and output functions. Although the DeepONets are considered to be resolution-agnostic with regard to the output function that can be evaluated at any one point at each inference, the input to the branch network is still constrained to a particular discretization. Furthermore, al-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

though adjustments can be made to convolution-based architectures to accept inputs of any size, they are typically not 'resolution-independent', in that the accuracy suffers significantly if tested on resolutions unseen during training (Li et al., 2020).

Neural operators (Li et al., 2020; Kovachki et al., 2022) are a class of resolution-independent neural network-based operator models for PDE problems. One of these architectures, the Fourier Neural Operator (FNO) (Li et al., 2021) has gained popularity in such fields as weather modeling (Kurth et al., 2023; Bonev et al., 2023). However, neural operators have been primarily explored for domain-to-domain mappings where the input is a function over the entire domain (typically an initial condition or material properties), and the output is over the entire domain as well (typically the solution at a later time). Despite the popularity of neural operators, their use to predict solution functions over a domain given only data over the boundary (such as a spatially varying Dirichlet boundary condition) remains unexplored.In this paper, we refer to such a problem as a boundary-to-domain (B2D) problem. This type of problems has a wide spectrum of applications in areas such as fluid mechanics, solid mechanics, heat transfer etc.

To address the challenge, we aim to develop a Fourier neural operator (FNO) based architecture for a boundary-to-domain problem in PDE simulation. In other words, we ask the question: how can we predict the solution over the domain, given a function over the boundary (a boundary condition) as input? To this end, we propose a novel FNO-based architecture, named *Lifting Product-FNO* (or LP-FNO) which can map arbitrary boundary functions defined on lower-dimensional manifolds to the entire solution domain. In particular, we extract hidden representations from two different FNOs defined on the lower-dimensional boundary domains, which are then lifted into the domain using a novel lifting product layer. For a simple, 1D boundary to a 2D domain problem, this lifting product operation is simply an outer product. In this paper, we demonstrate the B2D problem for the two-dimensional (2D) Poisson equation with non-homogeneous spatially-varying Dirichlet boundary conditions. We demonstrate the efficacy and resolution-independence of our proposed LP-FNO on this problem.

2. Boundary-to-Domain problem for PDEs

We define the boundary to domain problem for PDEs in this section. Suppose $\Omega \subset \mathbb{R}^d$ represents a region in d-dimensional space (usually, $d = 1, 2, 3$) with boundary $\partial\Omega$. An arbitrary PDE can be represented as

$$\boldsymbol{R}(\boldsymbol{u}(\boldsymbol{x}, t)) = \boldsymbol{0}, \qquad \boldsymbol{x} \in \Omega,\ t \in [0, T], \tag{1}$$

with boundary conditions

$$\boldsymbol{b}(\boldsymbol{u}(\boldsymbol{x}, t)) = \boldsymbol{0} \qquad \boldsymbol{x} \in \partial\Omega,\ t \in [0, T] \tag{2}$$

where $\boldsymbol{u} \in C([0, T], L^2(\Omega))^m$, the state, is a set of functions defined on the domain of interest ($\Omega \times [0, T]$) in space-time, $\boldsymbol{R}$ is a linear or nonlinear differential operator on the space of states, and $\boldsymbol{b}$ is an operator on the space of traces of the solution on the boundary $\partial\Omega$ of the domain. Steady-state scalar PDEs are those that do not depend on time and have a single variable of interest $u \in L^2(\Omega)$. An example is the Dirichlet problem for the Poisson equation:

$$-\nabla^2 u(\boldsymbol{x}) = f, \qquad \boldsymbol{x} \in \Omega, \tag{3}$$

$$u(\boldsymbol{x}) = g \qquad \boldsymbol{x} \in \partial\Omega. \tag{4}$$

In this paper, we define the boundary-to-domain problem for PDEs as one of predicting the n-dimensional solution functions that obeys the PDE, given $(n-1)$-dimension functions over the boundary that represent the boundary conditions. Our aim is to learn the map

$$\mathcal{G} : D \subset L_2(\partial\Omega) \to L_2(\Omega). \tag{5}$$

In the case of the Poisson problem, this would map the boundary function g in equation (4) to the solution u.

3. Lifting Product Fourier Neural Operator

Let $\boldsymbol{g} : \Gamma \to \mathbb{R}^m$ be the input boundary function for m physical variables (where $\Gamma \subset \partial\Omega$). The schematic of our proposed Lifting-Product FNO (LP-FNO) is illustrated in figure 1, where we generate two separate feature representations of the input function. These are generated by two FNO blocks, each of which takes the boundary function $\boldsymbol{g}$ as the input.

FNO involves a hidden embedding space of dimension n_e (Li et al., 2021). Assuming $\boldsymbol{g}$ is discretized by N points, initially, hidden representations $\boldsymbol{h} \in \mathbb{R}^{N \times n_e}$ are generated by a learned point-wise lifting operator $\boldsymbol{Q} : \mathbb{R}^m \to \mathbb{R}^{n_e}$, not to be confused with the lifting product described later in this section:

$$\boldsymbol{h}_0 = \boldsymbol{Q}\boldsymbol{g}. \tag{6}$$

An FNO block consists of several FNO layers. Each FNO layer $\boldsymbol{F}_{\theta_i} : \mathbb{R}^{N \times n_e} \to \mathbb{R}^{N \times n_e}$ parameterized using θ_i for $i \in \{1, 2\}$ is given by (Li et al., 2021)

$$\boldsymbol{F}_{\theta_i}(\boldsymbol{h}) = \sigma(\mathcal{F}^{-1}(\boldsymbol{R}_i(\mathcal{F}(\boldsymbol{h}))) + \boldsymbol{W}_i\boldsymbol{h}) \tag{7}$$

$$\text{or,} \quad \boldsymbol{F}_{\theta_i} = \sigma(\mathcal{F}^{-1}\boldsymbol{R}_i\mathcal{F} + \boldsymbol{W}_i) \tag{8}$$

where $\mathcal{F}$ is the lower-dimensional real Fourier transform. $\boldsymbol{R}$ is a learned linear operator that acts independently on each mode of the Fourier transform but couples the different 'channels' of the n_e-dimensional embedding space.

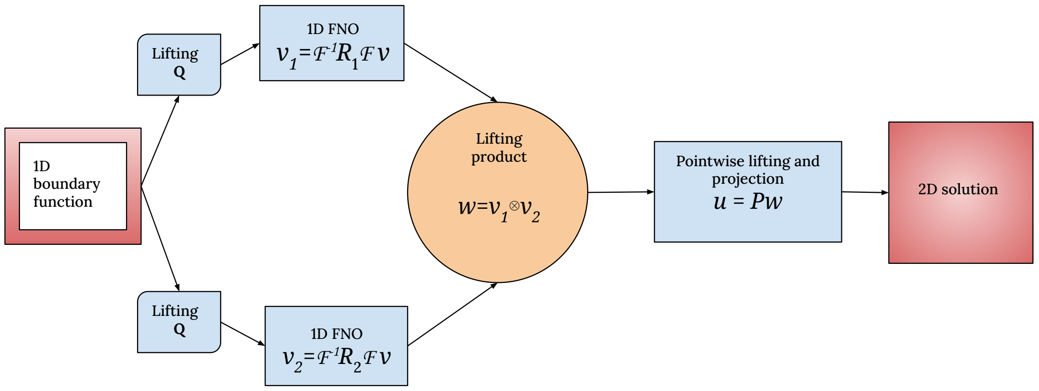
1D boundary function
Lifting Q
Lifting Q
1D FNO $v_1 = \mathcal{F}^{-1} R_1 \mathcal{F} v$
1D FNO $v_2 = \mathcal{F}^{-1} R_2 \mathcal{F} v$
Lifting product $w = v_1 \otimes v_2$
Pointwise lifting and projection $u = Pw$
2D solution

Figure 1. A schematic representation of the LP-FNO architecture example for the 1D to 2D case

$\boldsymbol{W} : \mathbb{R}^{n_e} \to \mathbb{R}^{n_e}$ is a point-wise linear neural network layer and σ applies a scalar nonlinear activation to each hidden channel at each grid point. We observe that the operations responsible for globally coupling all points in the spatial grid are the forward and inverse Fourier transforms.

Let $\boldsymbol{P}$ be a point-wise linear projection down from the embedding space to the state space of the PDE. Then LP-FNO, with l layers of lower-dimensional FNO in each FNO block, can be written as

$$\boldsymbol{v}_i = \prod_{j=1}^{l} (\boldsymbol{F}_{\theta_{ij}}) \boldsymbol{Q}\, \boldsymbol{g} \quad i = 1, 2; \tag{9}$$

$$\boldsymbol{u} = \boldsymbol{P}(\boldsymbol{v}_1 \otimes \boldsymbol{v}_2), \tag{10}$$

where $\prod$ denotes sequential composition of functions. Note that the lifting product $\otimes : L^2(\mathbb{R}^d) \times L^2(\mathbb{R}^d) \to L^2(\mathbb{R}^{d+1})$, to be defined below in the discrete sense, acts separately on each embedding dimension. $\boldsymbol{W}, \boldsymbol{Q}$ and $\boldsymbol{P}$ all act separately at each point in physical space using feedforward neural networks. The projection operator $\boldsymbol{P} : \mathbb{R}^{n_e} \to \mathbb{R}^m$ projects back to the space of physical variables.

Lifting Product: Let a and b be two (discretized) functions on the lower-dimensional boundary of a rectangular domain. The tensor operation that lifts the input to a higher dimensional function c can be written as

$$c_{ij} = a_i b_j \quad \text{in 1D to 2D lifting} \tag{11}$$

and

$$c_{kij} = a_{ij} b_{ik} \quad \text{in 2D to 3D lifting} \tag{12}$$

(no summation implied). The 1D to 2D operation is equivalent to an outer product $\boldsymbol{c} = \boldsymbol{a}\boldsymbol{b}^T$ separately on each channel. For the purpose of this paper, through some abuse of notation, we will denote both lifting product operations by $\boldsymbol{c} = \boldsymbol{a} \otimes \boldsymbol{b}$.

4. Experimental Setup

4.1. Baselines

Resolution-agnostic Tencoder: As a baseline, we use a modification of Tencoder (Kashi, 2023), which is a convolutional neural network-based encoder-decoder architecture with a tensor product layer in the decoder. While the architecture was designed to train and test only on one fixed resolution, we modify it to work with varying input and output resolutions. Our modifications include using an adaptive average pooling layer in the encoder to downsample to a fixed-size latent space independent of the input size, as well as the use of bilinear interpolation in the decoder which enables upsampling to the required target output size. This enables the model trained on an arbitrary resolution to predict solutions for boundary functions sampled on uniform grids of different sizes, not just the one(s) it was trained on.

FNO with zero padding: We use a FNO-2d architecture (Li et al., 2021). However, that architecture was designed for a domain-to-domain scenario and requires a function over the entire domain as input. To address this, the input to the FNO is the 1D-boundary function g padded with zeros all over the 2D-domain Ω.

4.2. Problem setup

We solve the boundary-to-domain problem for the Poisson equation (3) with Dirichlet conditions (equation (4)). The problem setup is identical to the one used to evaluate Tencoder (Kashi, 2023). In summary, three families of boundary functions are specified on the left boundary of a square domain and simulations run to obtain training and test datasets. The boundary functions are parameterized by a few parameters (this parameterization is never exposed to the model), and there are separate test sets containing in-distribution and out-of-distribution samples with respect to the training set. The training set consists of 2048 samples

		In-Distribution		Out-of-Distribution					
				Gaussian		Sinusoidal		Polynomial	
Resolution	**Model**	**Rel. L1**	**Rel L2**	**Rel. L1**	**Rel L2**	**Rel. L1**	**Rel L2**	**Rel. L1**	**Rel L2**
32	Tencoder	0.01257	0.00426	0.90621	0.40405	1.74984	0.94939	0.91356	0.44197
	FNO2d	0.00254	0.00139	0.73337	0.39078	1.11024	0.73673	0.71	0.3975
	TP-FNO	0.0108	0.00463	0.30713	0.05389	0.52667	0.15851	0.3259	0.06815
64	Tencoder	0.04765	0.0238	0.03826	0.02354	0.23802	0.15035	0.05085	0.02518
	FNO2d	0.00541	0.0019	0.00347	0.00126	0.06382	0.0191	0.0142	0.00435
	TP-FNO	0.01937	0.0064	0.00542	0.0016	0.19085	0.0889	0.04098	0.01268
128	Tencoder	0.15391	0.0682	1.14888	0.53499	2.21484	1.2091	1.15768	0.56814
	FNO2d	0.01125	0.00271	2.07062	1.376	2.15543	1.6864	1.96741	1.3438
	TP-FNO	0.03501	0.0082	0.05156	0.01847	0.3278	0.13745	0.111	0.0374

Table 1. Relative L1 and L2 norm errors of predictions from four models trained on data at different resolutions. In the in-distribution (I.D.) test data are at each model's 'native' resolution (the one it was trained on) while the out-of-distribution (O.O.D.) test data are at a 64x64 resolution for all the models in this table.

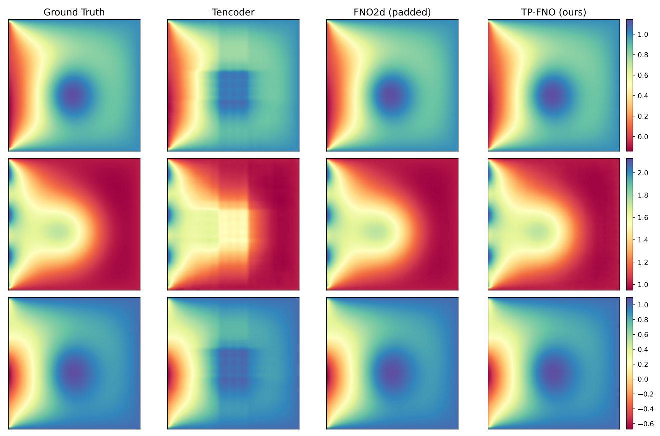

Figure 2. Comparison of the predicted solutions on in-distribution examples for models trained on 64x64 resolution.

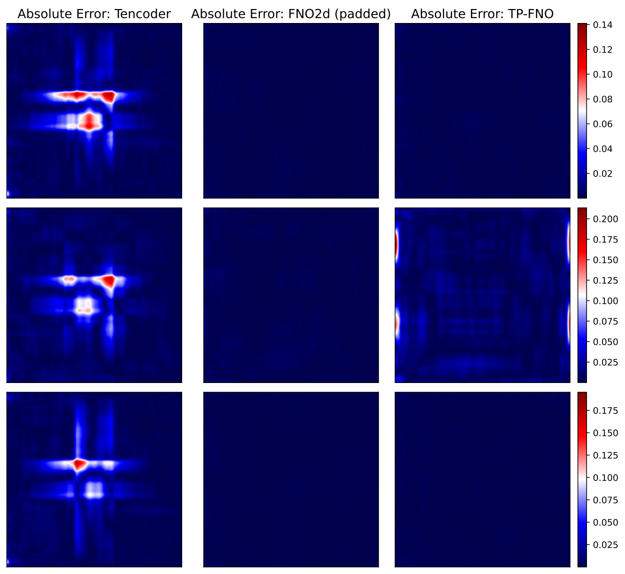

Figure 3. Comparison of the absolute error on in-distribution examples for models trained on 64x64 resolution.

(pairs of boundary condition and corresponding solution) for two families of boundary functions - Gaussian and sinusoidal. We test on in-distribution and out-of-distribution samples from those two families but also from a third family of polynomial functions up to degree 4. We train the models with data on some fixed resolution, and test it on data on both the same resolution ("native resolution") and also other resolutions it has not seen during training ("non-native resolutions").

5. Results

In Table 1, we first show results from the three architectures on in-distribution samples on the same resolution they were trained on, and out-of-distribution (OOD) samples on a 64x64 grid. All the models were trained for 200 epochs over the training set comprising 2048 samples.

In-distribution performance of LP-FNO with baselines: On the in-distribution test samples, we observe that the FNO2d (padded with zeros) and our proposed LP-FNO shows similar accuracy in around 10^{-3} relative L2 errors, with FNO-2d having a slight edge in performance for resolutions 64×64 and 128×128. The Tencoder consistently performs significantly worse that the other two models, with almost an order of magnitude difference in performance. Three example predictions of the solutions from the test set and their respective absolute errors are shown in Figure 2 and Figure 3 respectively. Similar to our previous observation, we see that both the PDE solutions and the absolute errors for the FNO-2d and LP-FNO are comparable,

Training Resolution	**Model**	**Testing Resolution**		
		32	**64**	**128**
32	Tencoder	0.00426	0.34346	0.3282
	FNO2d	0.00139	0.29973	0.3535
	TP-FNO	0.00463	0.05119	0.09289
64	Tencoder	0.41229	0.0238	0.32779
	FNO2d	0.6804	0.0019	0.58979
	TP-FNO	0.05879	0.0064	0.04006
128	Tencoder	0.46239	0.42954	0.0682
	FNO2d	2.29772	1.17268	0.00271
	TP-FNO	0.09778	0.02646	0.0082

Table 2. Relative L2 norms of error on resolution-independence on in-distribution test sets.

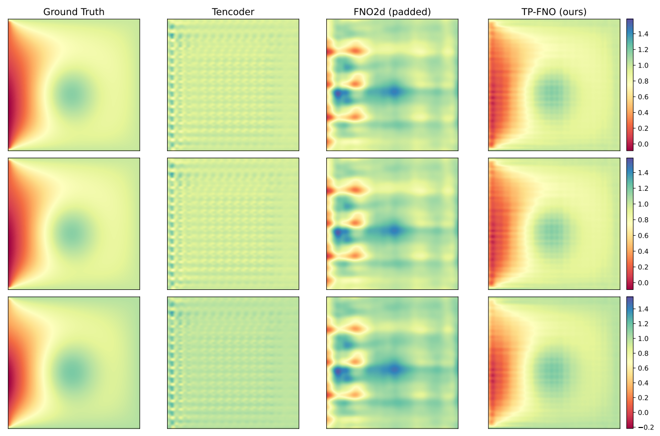

Figure 4. Comparison of the predicted solutions on out-of-distribution exponential examples for models trained on 32x32 resolution and evaluated on 64x64.

while the predictions of Tencoder demonstrate checkerboard artefacts which are also visible in the absolute error plots. See more visualization of in-distribution examples in the Appendix.

Out-of-Distribution Generalization of LP-FNO: However, on out-of-distribution test sets on non-native resolution of 64x64 (the first and third row-blocks), padded FNO2d performs poorly, with relative errors greater than 1. In comparison, LP-FNO still retains a fair accuracy, with eg., 1.8% accuracy in relative L2 error with a model that is trained on 128x128 data and inferenced on 64x64 Gaussian samples. In that example, padded 2D FNO has a huge error rate of 137%. While this result for the FNO2d needs to be investigated further and corroborated, it points to a loss of resolution-independence due to the zero-padding required in input processing. As expected, the resolution-agnostic Tencoder, while being able to operate seamlessly on inputs of any resolution, does not perform well on non-native resolutions.

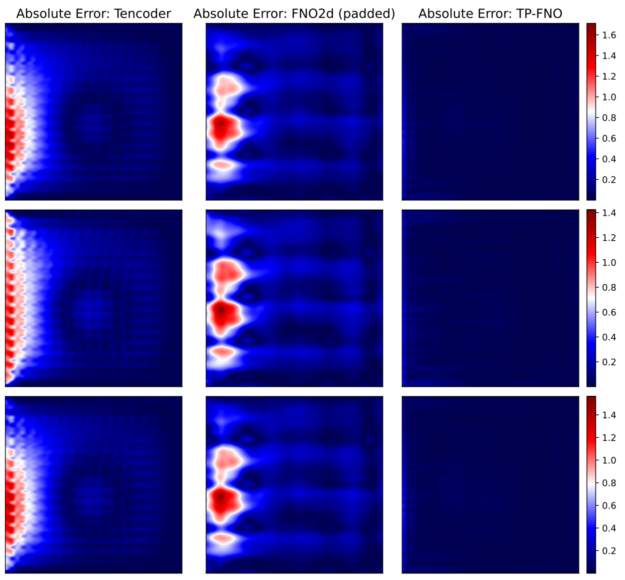

Figure 5. Comparison of the absolute error on out-of-distribution exponential examples for models trained on 32x32 resolution and evaluated on 64x64.

Figure 4 and 5 present the out-of-distribution solution fields and absolute errors of models trained on 32×32 and inferenced on out-of-distribution 64-dimensions inputs from the Gaussian family. We observe that the predictions for both the Tencoder and the FNO2d are extremely poor. While, the solutions obtained by of our proposed LP-FNO has some artefacts, it was able to capture some of the high-level patterns present in the solution. These high-level similarities in the solution demonstrate the resolution-independence capabilities of the LP-FNO, on out-of-distribution sample, which is a challenging task. Further, we believe that these artefacts shown by LP-FNO can be improved with better hyperparameter tuning and modifications to the LP-FNO model architecture as detailed in section 6. See more visualization of out-of-distribution examples for sinusoidal and polynomial cases in the appendix.

Resolution Independence of LP-FNO: Table 2 shows the degree to which each of the architectures is resolution-independent using test sets which are in-distribution in terms of boundary function parameters. Here, again, we see that on native resolutions (diagonal blocks in the table), all of the three models are able to learn good solutions, with FNO2d and TP-FNO being approximately an order of magnitude better than Tencoder. However, when tested on non-native resolution, the resolution independence fails to hold for Tencoder and FNO2d and we observe very large L2 errors. In contrast, LP-FNO demonstrates a reasonable accuracy even on non-native resolutions. We also visualize

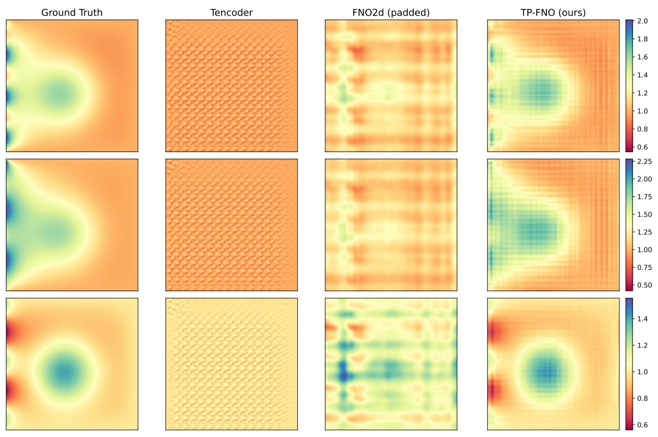

Figure 6. Comparison of the predicted solutions for models trained on 32x32 resolution and evaluated on 128x128.

the zero-shot super-resolution capabilities of the three models in Figure 6, where the models are trained on 32×32 and then evaluated on 128×128. As expected, we observe that the Tencoder and FNO2d fails to perform zero-shot super-resolution, while our proposed LP-FNO is able to capture the high-level details of the solution functions. However, it must be noted that LP-FNO still shows significant checkerboard artefacts in the solutions, and our future work would focus on that.

6. Conclusion and Future Work

We have presented LP-FNO, a neural operator based on FNO and a lifting product operation and demonstrated its potential on a simple 2D model problem. While our architecture still demonstrates some artefacts in its solution for out-of-domain and resolution independence tasks, it generalizes well to other resolutions, which puts it on the right track in terms of fulfilling the promise of neural operators for boundary-to-domain problems in PDE surrogate models.

The architecture presented is still a preliminary work part of ongoing developments. In the near future, several improvements are planned which are expected to significantly improve accuracy and efficiency on both native and non-native resolutions.

- Our aim is to implement LP-FNO with tensor product in modal (or frequency) space rather than physical space. We expect this to make the architecture more scalable while improving its accuracy.
- We will investigate the effect of adding 2D FNO layers after the tensor product layer in addition to the simple linear projection layer used in the current implementation.
- We will test the architecture on nonlinear PDEs and experiment with different activation functions.
- We intend to prove theoretically that our architecture has the universal approximation property in a relevant function space and that it is resolution independent.

7. Impact statement

This paper presents work whose goal is to advance the field of machine learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here. The eventual most direct impact will be in achieving transformational acceleration of computer simulations in science and engineering.

References

Bonev, B., Kurth, T., Hundt, C., Pathak, J., Baust, M., Kashinath, K., and Anandkumar, A. Spherical Fourier neural operators: Learning stable dynamics on the sphere. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 2806–2823. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/bonev23a.html.

Chen, Y., Lu, L., Karniadakis, G. E., and Negro, L. D. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Optics Express*, 28(8): 11618–11633, April 2020. ISSN 1094-4087. doi: 10.1364/OE.384875. Publisher: Optica Publishing Group.

Kashi, A. Tencoder: tensor-product encoder-decoder architecture for predicting solutions of pdes with variable boundary data. In *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, SC-W '23, pp. 102–108, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400707858. doi: 10.1145/3624062.3626088.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Learning maps between function spaces, October 2022. arXiv:2108.08481 [cs, math].

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 26548–26560. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/df438e5206f31600e6ae4af72f2725f1-Paper.pdf.

Kurth, T., Subramanian, S., Harrington, P., Pathak, J., Mardani, M., Hall, D., Miele, A., Kashinath, K., and Anandkumar, A. Fourcastnet: Accelerating global high-resolution weather forecasting using adaptive fourier neural operators. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, PASC '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701900. doi: 10.1145/3592979.3593412.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Graph kernel network for partial differential equations, March 2020. arXiv:2003.03485 [cs, math, stat].

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations, 2021.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, mar 2021. doi: 10.1038/s42256-021-00302-5.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, February 2019. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.10.045.

A. Additional results

A.1. Visualization of Other In-Distribution Results

Additional visualizations of the model performances on 32x32 can be seen in Figures 7 and 8.

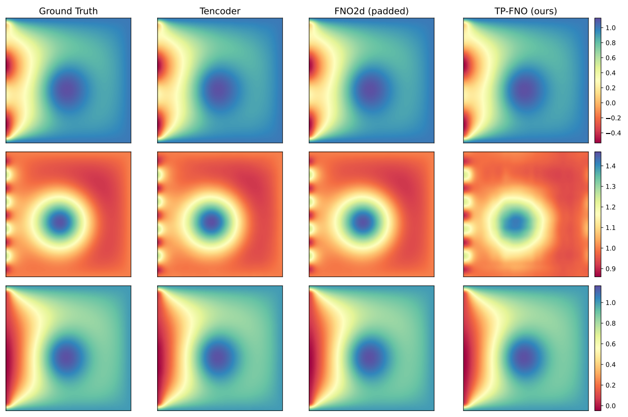

Figure 7. Comparison of the predicted solutions on in-distribution examples for models trained on 32x32 resolution.

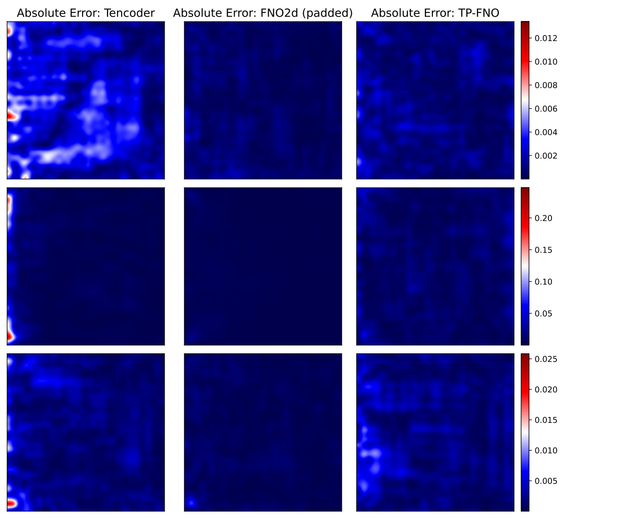

Figure 8. Comparison of the absolute error on in-distribution examples for models trained on 32x32 resolution.

Additional visualizations of the model performances on 128x128 can be seen in Figures 9 and 10.

A.2. Visualization of Other Out-of-Distribution Results

The out-of-distribution for the sinusoidal examples are shown in 11 and 12.

The out-of-distribution for the polynomial examples are shown in 13 and 14.

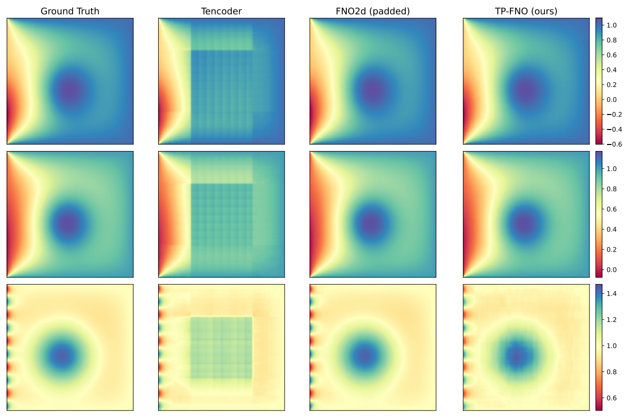

Figure 9. Comparison of the predicted solutions on in-distribution examples for models trained on 128x128 resolution.

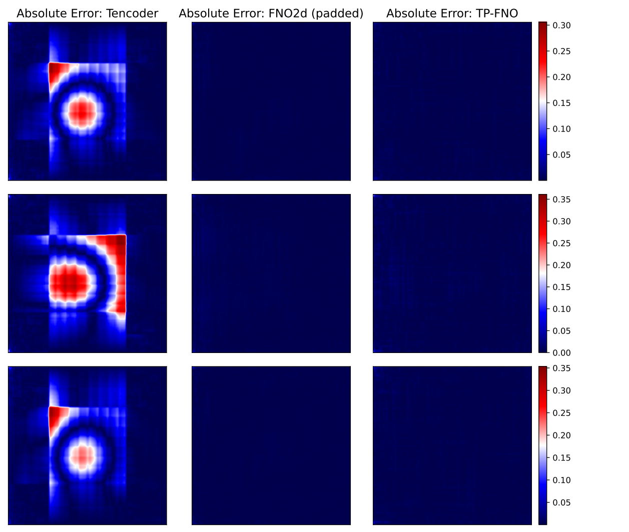

Figure 10. Comparison of the absolute error on in-distribution examples for models trained on 128x128 resolution.

A.3. Convergence of Different Models

Figure 15, 16, 17 shows the convergence of the mean squared error (MSE) loss of the different models while training on the 32x32, 64x64 and 128x128 datasets respectively.

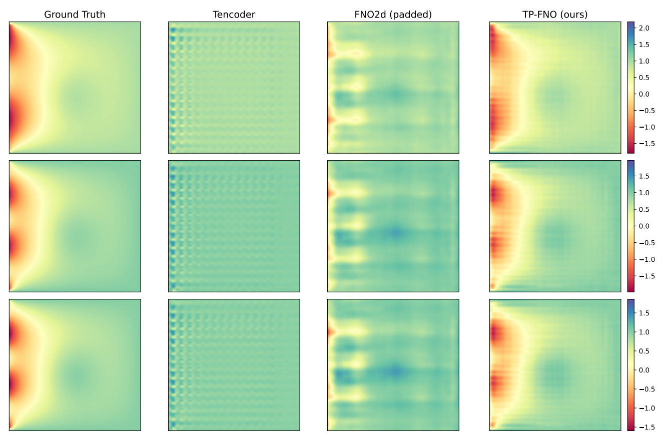

Figure 11. Comparison of the predicted solutions on out-of-distribution sinusoidal examples for models trained on 32x32 resolution and evaluated on 64x64.

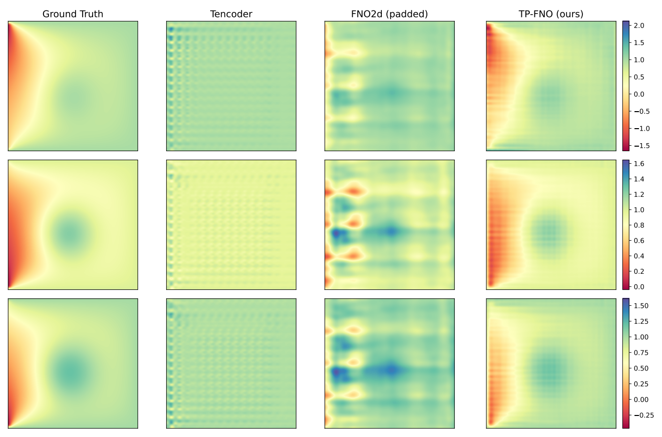

Figure 13. Comparison of the predicted solutions on out-of-distribution polynomial examples for models trained on 32x32 resolution and evaluated on 64x64.

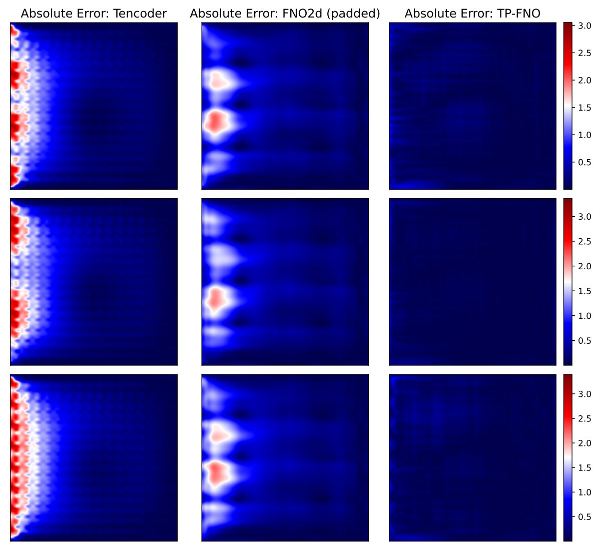

Figure 12. Comparison of the absolute error on out-of-distribution sinusoidal examples for models trained on 32x32 resolution and evaluated on 64x64.

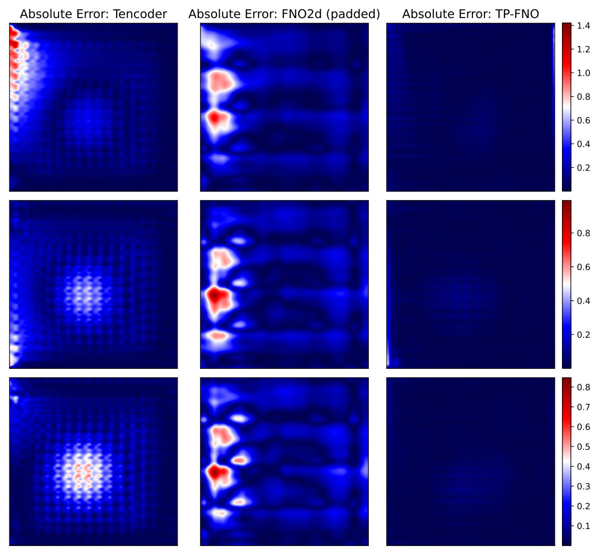

Figure 14. Comparison of the absolute error on out-of-distribution polynomial examples for models trained on 32x32 resolution and evaluated on 64x64.

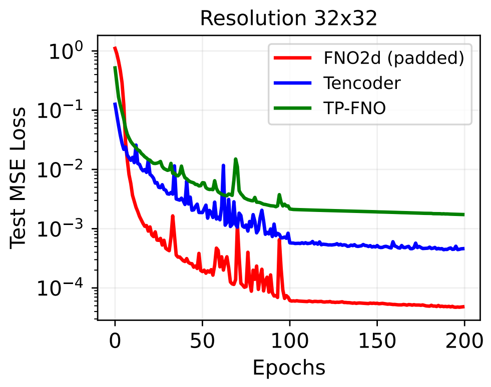

Figure 15. Test MSE loss during training on data of resolution 32x32

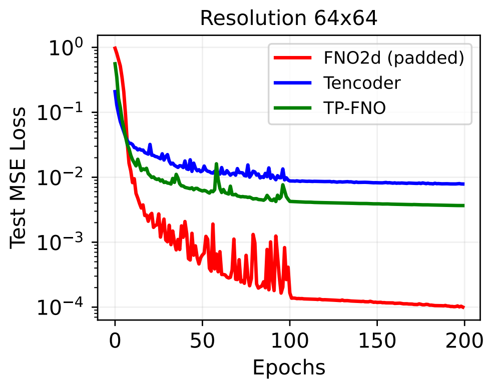

Figure 16. Test MSE loss on data of resolution 64x64

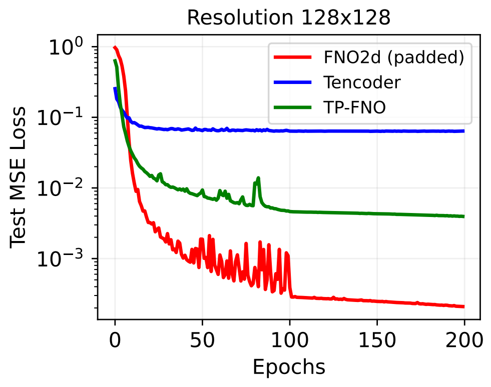

Figure 17. Test MSE loss on data of resolution 128x128