Domain Agnostic Fourier Neural Operators

Ning Liu[*]
Global Engineering and Materials, Inc.
Princeton, NJ 08540
ningliu@umich.edu

Siavash Jafarzadeh[*]
Department of Mathematics
Lehigh University
Bethlehem, PA 18015
sij222@lehigh.edu

Yue Yu[†]
Department of Mathematics
Lehigh University
Bethlehem, PA 18015
yuy214@lehigh.edu

Abstract

Fourier neural operators (FNOs) can learn highly nonlinear mappings between function spaces, and have recently become a popular tool for learning responses of complex physical systems. However, to achieve good accuracy and efficiency, FNOs rely on the Fast Fourier transform (FFT), which is restricted to modeling problems on rectangular domains. To lift such a restriction and permit FFT on irregular geometries as well as topology changes, we introduce domain agnostic Fourier neural operator (DAFNO), a novel neural operator architecture for learning surrogates with irregular geometries and evolving domains. The key idea is to incorporate a smoothed characteristic function in the integral layer architecture of FNOs, and leverage FFT to achieve rapid computations, in such a way that the geometric information is explicitly encoded in the architecture. In our empirical evaluation, DAFNO has achieved state-of-the-art accuracy as compared to baseline neural operator models on two benchmark datasets of material modeling and airfoil simulation. To further demonstrate the capability and generalizability of DAFNO in handling complex domains with topology changes, we consider a brittle material fracture evolution problem. With only one training crack simulation sample, DAFNO has achieved generalizability to unseen loading scenarios and substantially different crack patterns from the trained scenario. Our code and data accompanying this paper are available at `https://github.com/ningliu-iga/DAFNO`.

1 Introduction

Deep learning surrogate models provide a useful data-driven paradigm to accelerate the PDE-solving and calibration process of scientific modeling and computing problems. Among others, a wide range of scientific computing applications entail the learning of solution operators, i.e., the learning of infinite dimensional function mappings between any parametric dependence to the solution field. A prototypical instance is the case of solving Navier-Stokes equations in fluid mechanics, where the initial input needs to be mapped to a temporal sequence of nonlinear parametric solutions. The demand for operator learning has sparked the development of neural operator based methods (Li et al., 2020a,b,c; You et al., 2022a,b,c; Goswami et al., 2022; Liu et al., 2022; Gupta et al., 2021; Lu et al., 2019; Cao, 2021; Hao et al., 2023; Li et al., 2022b; Yin et al., 2022c), with one of the most popular architectures being Fourier Neural Operators (FNOs) (Li et al., 2020c; You et al., 2022c).

The success of FNOs can be mostly attributed to its convolution-based integral kernel that learns in a resolution-invariant manner and the computationally efficient evaluation achieved via Fast Fourier Transform (FFT) (Brigham, 1988). While learning in the spectral domain is fast, the latter comes at a cost: the computational domain of the underlying problem needs to be rectangular with uniformly

[*]Equal contribution
[†]Corresponding author

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

meshed grids. This is often intractable as the domain of interest is, more often than not, irregular. An often taken trick for applying FNO to irregular domains is to embed the original domain into a larger rectangular domain and zero-pad or extrapolate on the redundant space (Lu et al., 2022). This poses two potential problems, one being the possible numerical errors and even instabilities due to the discontinuity at the original domain boundary (e.g., the Gibbs phenomenon (Gottlieb & Shu, 1997)) and the other, perhaps more importantly, being the fact that the padding/extrapolating techniques cannot handle domains with shallow gaps, as is the case in object contact and crack propagation problems. Meanwhile, another line of work emphasizes the learning of a diffeomorphic mapping between the original domain and a latent domain with uniform grids on which FNO can be applied (Li et al., 2022a). However, in this approach the changes on the boundary and the domain topology can only be informed via the learned diffeomorphism, which results in approximation errors when tested on a new domain geometry and possible failure when a change in topology is involved on the domain geometry.

In this work, we aim to design FNO architectures that explicitly embed the boundary information of irregular domains, which we coin Domain Agnostic Fourier Neural Operator (DAFNO). This is inspired by the recent work in convolution-based peridynamics (Jafarzadeh et al., 2022b) in which bounded domains of arbitrary shapes are explicitly encoded in the nonlocal integral formulation. We argue that, by explicitly embedding the domain boundary information into the model architecture, DAFNO is able to learn the underlying physics more accurately, and the learnt model is generalizable to changes on the domain geometry and topology. Concretely, we construct two practical DAFNO variants, namely, eDAFNO that inherits the explicit FNO architecture (Li et al., 2022a) and iDAFNO that is built upon the implicit FNO (IFNO) architecture characterizing layer-independent kernels (You et al., 2022c). Moreover, a boundary smoothening technique is also proposed to resolve the Gibbs phenomenon and retain the fidelity of the domain boundary. In summary, the primary contributions of the current work are as follows:

- We propose DAFNO, a novel Fourier neural operator architecture that explicitly encodes the boundary information of irregular domains into the model architecture, so that the learned operator is aware of the domain boundary, and generalizable to different domains of complicated geometries and topologies.
- By incorporating a (smoothened) domain characteristic function into the integral layer, our formulation resembles a nonlocal model, such that the layer update acts as collecting interactions between material points inside the domain and cuts the non-physical influence outside the domain. As such, the model preserves the fidelity of the domain boundary as well as the convolution form of the kernel that retains the computational efficiency of FFT.
- We demonstrate the expressivity and generalizability of DAFNO across a wide range of scientific problems including constitutive modeling of hyperelastic materials, airfoil design, and crack propagation in brittle fracture, and show that our learned operator can handle not only irregular domains but also topology changes over the evolution of the solution.

2 Background and related work

The goal of this work is to construct a neural network architecture to learn common physical models on various domains. Formally, given $\mathcal{D} := \{(\boldsymbol{g}_i|_{\Omega_i}, \boldsymbol{u}_i|_{\Omega_i})\}_{i=1}^N$, a labelled set of function pair observations both defined on the domain $\boldsymbol{x} \in \Omega_i \subset \mathbb{R}^s$. We assume that the input $\{\boldsymbol{g}_i(\boldsymbol{x})\}$ is a set of independent and identically distributed (i.i.d.) random fields from a known probability distribution μ on $\mathcal{A}(\mathbb{R}^{d_g})$, a Banach space of functions taking values in $\mathbb{R}^{d_g}$. $\boldsymbol{u}_i(\boldsymbol{x}) \in \mathcal{U}(\mathbb{R}^{d_u})$, possibly noisy, is the observed corresponding response taking values in $\mathbb{R}^{d_u}$. Taking mechanical response modeling problem for example, Ω_i is the shape of the object of interest, $\boldsymbol{g}_i(\boldsymbol{x})$ may represent the boundary, initial, or loading conditions, and $\boldsymbol{u}_i(\boldsymbol{x})$ can be the resulting velocity, pressure, or displacement field of the object. We assume that all observations can be modeled by a common and possibly unknown governing law, e.g., balance laws, and our goal is to construct a surrogate operator mapping, $\tilde{\mathcal{G}}$, from $\mathcal{A}$ to $\mathcal{U}$ such that

$$\tilde{\mathcal{G}}[\boldsymbol{g}_i; \theta](\boldsymbol{x}) \approx \boldsymbol{u}_i(\boldsymbol{x}), \ \forall \boldsymbol{x} \in \Omega_i. \tag{1}$$

Here, θ represents the (trainable) network parameter set.

In real-world applications, the domain Ω_i can possess different topologies, e.g., in contact problems (Benson & Okazawa, 2004; Simo & Laursen, 1992) and material fragmentation problems (De Luycker et al., 2011; Agwai et al., 2011; Silling, 2003), and/or evolve with time, as is the case

in large-deformation problems (Shadden et al., 2010) and fluid–structure interaction applications (Kuhl et al., 2003; Kamensky et al., 2017). Hence, it is desired to develop an architecture with generalizability across various domains of complex shapes, so that the knowledge obtained from one geometry can be transferable to other geometries.

2.1 Learning solution operators of hidden physics

In real-world physical problems, predicting and monitoring complex system responses are ubiquitous in many applications. For these purposes, physics-based PDEs and tranditional numerical methods have been commonly employed. However, traditional numerical methods are solved for specific boundary, initial, and loading conditions $\boldsymbol{g}$ on a specific domain Ω. Hence, the solutions are not generalizable to other domains and operating conditions.

To provide a more efficient and flexible surrogate model for physical response prediction, there has been significant progress in the development of deep neural networks (NNs) and scientific machine learning models (Ghaboussi et al., 1998, 1991; Carleo et al., 2019; Karniadakis et al., 2021; Zhang et al., 2018; Cai et al., 2022; Pfau et al., 2020; He et al., 2021; Besnard et al., 2006). Among others, neural operators (Li et al., 2020a,b,c; You et al., 2022a; Ong et al., 2022; Gupta et al., 2021; Lu et al., 2019, 2021b; Goswami et al., 2022; Tripura & Chakraborty, 2022) show particular promise in learning physics of complex systems: compared with classical NNs, neural operators are resolution independent and generalizable to different input instances. Therefore, once the neural operator is trained, solving for a new instance of the boundary/initial/loading condition with a different discretization only requires a forward pass. These advantages make neural operators a useful tool to many physics modeling problems (Yin et al., 2022a; Goswami et al., 2022; Yin et al., 2022b; Li et al., 2020a,b,c; Lu et al., 2022, 2021a).

2.2 Neural operator learning

Here, we first introduce the basic architecture of the general integral neural operators (Li et al., 2020a,b,c; You et al., 2022a,c), which are comprised of three building blocks. First, the input function, $\boldsymbol{g}(\boldsymbol{x}) \in \mathcal{A}$, is lifted to a higher-dimensional representation via $\boldsymbol{h}^0(\boldsymbol{x}) = \mathcal{P}[\boldsymbol{g}](\boldsymbol{x}) := P[\boldsymbol{x}, \boldsymbol{g}(\boldsymbol{x})]^T + \boldsymbol{p}$, where $P \in \mathbb{R}^{(s+d_g)\times d_h}$ and $\boldsymbol{p} \in \mathbb{R}^{d_h}$ define an affine pointwise mapping. Then, the feature vector function $\boldsymbol{h}^0(\boldsymbol{x})$ goes through an iterative layer block, where the layer update is defined via the sum of a local linear operator, a nonlocal integral kernel operator, and a bias function: $\boldsymbol{h}^{l+1}(\boldsymbol{x}) = \mathcal{J}^{l+1}[\boldsymbol{h}^l](\boldsymbol{x})$. Here, $\boldsymbol{h}^l(\boldsymbol{x}) \in \mathbb{R}^{d_h}$, $l = 0, \cdots, L$, is a sequence of functions representing the values of the network at each hidden layer. $\mathcal{J}^1, \cdots, \mathcal{J}^L$ are the nonlinear operator layers defined by the particular choice of networks. Finally, the output $\boldsymbol{u}(\boldsymbol{x}) \in \mathcal{U}$ is obtained via a projection layer by mapping the last hidden layer representation $\boldsymbol{h}^L(\boldsymbol{x})$ onto $\mathcal{U}$ as: $\boldsymbol{u}(\boldsymbol{x}) = \mathcal{Q}[\boldsymbol{h}^L](\boldsymbol{x}) := Q_2\sigma(Q_1\boldsymbol{h}^L(\boldsymbol{x}) + \mathbf{q}_1) + \mathbf{q}_2$. Q_1, Q_2, $\mathbf{q}_1$ and $\mathbf{q}_2$ are the appropriately sized matrices and vectors that are part of the learnable parameter set, and σ is an activation function (e.g., ReLU (He et al., 2018) or GeLU).

Then, the system response can be learnt by constructing a surrogate operator of (1): $\tilde{\mathcal{G}}[\boldsymbol{g};\theta](\boldsymbol{x}) := \mathcal{Q} \circ \mathcal{J}^1 \circ \cdots \circ \mathcal{J}^L \circ \mathcal{P}[\boldsymbol{g}](\boldsymbol{x}) \approx \boldsymbol{u}(\boldsymbol{x})$, by solving the network parameter set θ via an optimization problem:

$$\min_{\theta\in\Theta} \mathcal{L}_{\mathcal{D}}(\theta) := \min_{\theta\in\Theta} \sum_{i=1}^{N} [C(\tilde{\mathcal{G}}[\boldsymbol{g}_i;\theta], \boldsymbol{u}_i)] \,. \tag{2}$$

Here, C denotes a properly defined cost functional (e.g., the relative mean square error) on Ω_i.

2.3 Fourier neural operators

The Fourier neural operator (FNO) is first proposed in Li et al. (2020c) with its iterative layer architecture given by a convolution operator:

$$\mathcal{J}^l[\boldsymbol{h}](\boldsymbol{x}) := \sigma\left(W^l\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{c}^l + \int_\Omega \kappa(\boldsymbol{x}-\boldsymbol{y};\boldsymbol{v}^l)\boldsymbol{h}(\boldsymbol{y})d\boldsymbol{y}\right) , \tag{3}$$

where $W^l \in \mathbb{R}^{d_h\times d_h}$ and $\boldsymbol{c}^l \in \mathbb{R}^{d_h}$ are learnable tensors at the l-th layer, and $\kappa \in \mathbb{R}^{d_h\times d_h}$ is a tensor kernel function with parameters $\boldsymbol{v}^l$. When a rectangular domain Ω with uniform meshes is considered, the above convolution operation can be converted to a multiplication operation through discrete Fourier transform:

$$\mathcal{J}^l[\boldsymbol{h}](\boldsymbol{x}) = \sigma\left(W^l\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{c}^l + \mathcal{F}^{-1}[\mathcal{F}[\kappa(\cdot;\boldsymbol{v}^l)]\cdot\mathcal{F}[\boldsymbol{h}(\cdot)]](\boldsymbol{x})\right) ,$$

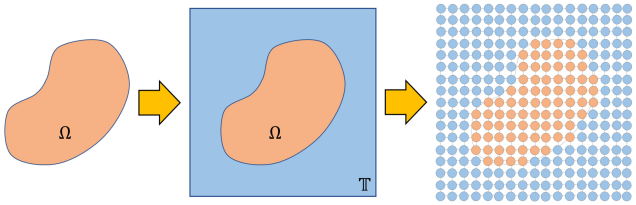

Figure 1: A schematic of extending an arbitrarily shaped domain Ω to a periodic box $\mathbb{T}$ and its discretized form in 2D.

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier transform and its inverse, respectively, which are computed using the FFT algorithm to each component of $\boldsymbol{h}$ separately. The FFT calculations greatly improve the computational efficiency due to their quasilinear time complexity, but they also restrict the vanilla FNO architecture to rectangular domains Ω (Lu et al., 2022).

To enhance the flexibility of FNO in modeling complex geometries, in Lu et al. (2022) the authors proposed to pad and/or extrapolate the input and output functions into a larger rectangular domain. However, such padding/extrapolating techniques are prone to numerical instabilities (Gottlieb & Shu, 1997), especially when the domain is concave and/or with complicated boundaries. As shown in Lu et al. (2022), the performance of dgFNO+ substantially deteriorates when handling complicated domains with notches and gaps. In Geo-FNO (Li et al., 2022a), an additional neural network is employed and trained from data, to continuously map irregular domains onto a latent space of rectangular domains. As a result, the vanilla FNO can be employed on the rectangular latent domain. However, this strategy relies on the continuous mapping from the physical domain to a rectangular domain, hence it is restricted to relatively simple geometries with no topology change.

3 Domain Agnostic Fourier Neural Operators

In this section, we introduce Domain Agnostic Fourier Neural Operator (DAFNO), which features the generalizability to new and unseen domains of arbitrary shapes and different topologies. The key idea is to explicitly encode the domain information in the design while retaining the convolutional architecture in the iterative layer of FNOs. In what follows, we present the eDAFNO architecture based on the standard/explicit FNO model (Li et al., 2020c), while the iDAFNO architecture based on the implicit FNO model (You et al., 2022c) is provided in Appendix A.

Concretely, we enclose the physical domain of interest, Ω, by a (slightly larger) periodic box $\mathbb{T}$, as shown in Figure 1. Next, we define the following domain characteristic function:

$$\chi(\boldsymbol{x}) = \begin{cases} 1 & \boldsymbol{x} \in \Omega \\ 0 & \boldsymbol{x} \in \mathbb{T} \setminus \Omega \end{cases}, \tag{4}$$

which encodes the domain information of different geometries. Inspired by Jafarzadeh et al. (2022b), we incorporate the above-encoded domain information into the FNO architecture of (3), by multiplying the integrand in its convolution integral with $\chi(\boldsymbol{x})\chi(\boldsymbol{y})$:

$$\mathcal{J}^l[\boldsymbol{h}] = \sigma\left(\int_{\mathbb{T}} \chi(\boldsymbol{x})\chi(\boldsymbol{y})\kappa(\boldsymbol{x}-\boldsymbol{y};\boldsymbol{v}^l)(\boldsymbol{h}(\boldsymbol{y})-\boldsymbol{h}(\boldsymbol{x}))d\boldsymbol{y} + W^l\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{c}^l\right). \tag{5}$$

Herein, we have followed the practice in You et al. (2022a) and reformulated (3) to a nonlocal Laplacian operator, which is found to improve training efficacy. By introducing the term $\chi(\boldsymbol{x})\chi(\boldsymbol{y})$, the integrand vanishes when either point $\boldsymbol{x}$ or $\boldsymbol{y}$ is positioned inside Ω and the other is positioned outside. This modification eliminates any undesired interaction between the regions inside and outside of Ω. As a result, it tailors the integral operator to act on Ω independently and is able to handle different domains and topologies. With this modification, the FFT remains applicable, since the convolutional structure of the integral is preserved and the domain of operation yet spans to the whole rectangular box $\mathbb{T}$. In this context, (5) can be re-organized as:

$$\mathcal{J}^l[\boldsymbol{h}] = \sigma\left(\chi(\boldsymbol{x})\Big(\int_{\mathbb{T}} \kappa(\boldsymbol{x}-\boldsymbol{y};\boldsymbol{v}^l)\chi(\boldsymbol{y})\boldsymbol{h}(\boldsymbol{y})d\boldsymbol{y} - \boldsymbol{h}(\boldsymbol{x})\int_{\mathbb{T}} \kappa(\boldsymbol{x}-\boldsymbol{y};\boldsymbol{v}^l)\chi(\boldsymbol{y})d\boldsymbol{y} + W^l\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{c}^l\Big)\right).$$

Note that multiplying $W^l\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{c}^l$ with $\chi(\boldsymbol{x})$ does not alter the computational domain. Now that the integration region is a rectangular box $\mathbb{T}$, the FFT algorithm and its inverse can be readily applied,

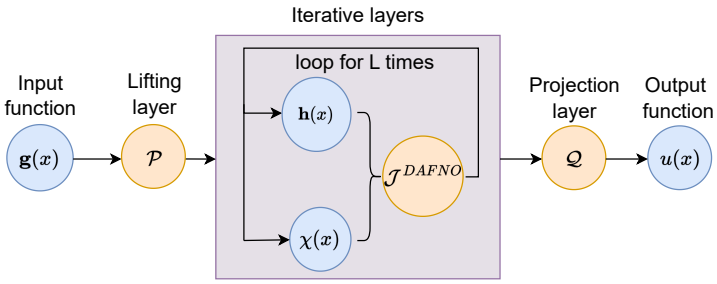

Figure 2: An illustration of the proposed DAFNO architecture. We start from the input function $\boldsymbol{g}(x)$. After lifting, the iterative Fourier layers are built that explicitly embed the encoded domain information, χ. Lastly, we project the last hidden layer representation to the target function space.

and hence we can further express the eDAFNO architecture as:

$$\begin{aligned} \mathcal{J}^l[\boldsymbol{h}] :=& \sigma\Big(\chi(\boldsymbol{x})\big(\mathcal{I}(\chi(\cdot)\boldsymbol{h}(\cdot);\boldsymbol{v}^l) - \boldsymbol{h}(\boldsymbol{x})\mathcal{I}(\chi(\cdot);\boldsymbol{v}^l) + W^l\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{c}^l\big)\Big) , \\ \text{where} \quad & \mathcal{I}(\circ;\boldsymbol{v}^l) := \mathcal{F}^{-1}\big[\mathcal{F}[\kappa(\cdot;\boldsymbol{v}^l)] \cdot \mathcal{F}[\circ]\big] . \end{aligned} \tag{6}$$

An illustration of the DAFNO atchitecture is provided in Figure 2. Note that this architectural modification is performed at the continuum level and therefore is independent of the discretization. Then, the box $\mathbb{T}$ can be discretized with structured grids (cf. Figure 1), as is the case in standard FNO.

Although the proposed DAFNO architecture in (6) can handle complex generalization in domains, it has a potential pitfall: since the characteristic function is not continuous on the domain boundary, its Fourier series cannot converge uniformly and the FFT result would present fictitious wiggling near the discontinuities (i.e., the Gibbs phenomenon (Day et al., 1965)). As a consequence, the introduction of χ can potentially jeopardize the computational accuracy. To improve the efficacy of DAFNO, we propose to replace the sharp characteristic function, χ, with a smoothed formulation:

$$\tilde{\chi}(\boldsymbol{x}) := \tanh(\beta \text{dist}(\boldsymbol{x}, \partial\Omega))(\chi(\boldsymbol{x}) - 0.5) + 0.5 . \tag{7}$$

Here, the hyperbolic tangent function $\tanh(z) := \frac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)}$, $\text{dist}(\boldsymbol{x}, \partial\Omega)$ denotes the (approximated) distance between $\boldsymbol{x}$ and the boundary of domain Ω, and β controls the level of smoothness, which is treated as a tunable hyperparameter. An illustration of the effect of the smoothed $\tilde{\chi}$ is displayed in Figure 3, with additional plots and prediction results with respect to different levels of smoothness, β, provided in Appendix B.1. In what follows, the $\tilde{\cdot}$ sign is neglected for brevity.

Remark: We point out that the proposed smoothed geometry encoding technique, although simple, is substantially different from existing function padding/extrapolation techniques proposed in Lu et al. (2022) who cannot handle singular boundaries (as in the airfoil tail of our example 2) nor notches/shallow gaps in the domain (as in the crack propagation of our example 3). Our proposed architecture in (5) is also more sophiscated than a trivial zero padding at each layer in that the characteristic function $\chi(\boldsymbol{x})$ is multiplied with the integrand, whereas the latter breaks the convolutional structure and hinders the application of FFTs.

4 Numerical examples

In this section, we demonstrate the accuracy and expressivity of DAFNO across a wide variety of scientific problems. We compare the performance of DAFNO against other relevant scientific machine learning models, including FNO (Li et al., 2020c), Geo-FNO (Li et al., 2022a), IFNO (You et al., 2022c), F-FNO (Tran et al., 2022), GNO (Li et al., 2020a), DeepONet (Lu et al., 2019), and UNet (Ronneberger et al., 2015). In particular, we carry out three experiments on irregular domains, namely, constitutive modeling of hyperelasticity in material science, airfoil design in fluid mechanics, and crack propagation with topology change in fracture mechanics. For fair comparison, the hyperparameters of each model are tuned to minimize the error on validation datasets, including initial learning rate, decay rate for every 100 epochs, smoothing parameter, and regularization parameter, while the total number of epochs is restricted to 500 for computational efficiency. The

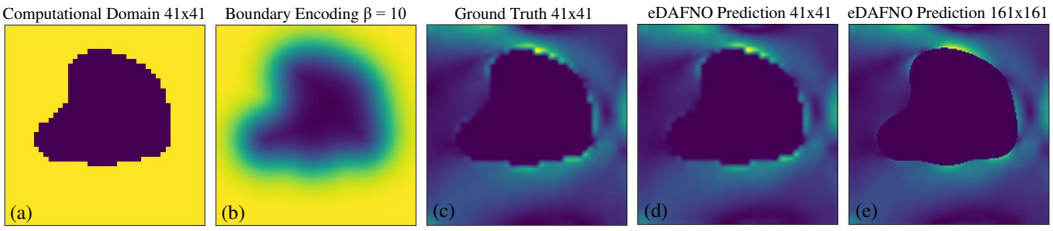

Figure 3: An illustration on a hyperelasticity sample: (a) sharp characteristic function χ, (b) smoothed characteristic function χ, (c) ground truth, (d) eDAFNO (trained using 41×41 discretization) prediction from the same resolution, and (e) zero-shot super-resolution prediction from eDAFNO (trained using 41×41 discretization and evaluated directly on 161×161 discretization).

relative L2 error is reported as comparison metrics for both the training and test datasets. The experiments of each method are repeated on 5 trials with 5 different random seeds, and the mean and standard deviation of the errors are reported. Further details and additional results are provided in Appendix B.

4.1 Constitutive modeling of hyperelasticity

We start with a hyperelasticity problem in material science that models the fundamental principle of constitutive relations. The high-fidelity synthetic data in this benchmark is governed by

$$\rho\frac{\partial^2 \boldsymbol{u}}{\partial t^2} + \nabla\cdot\boldsymbol{\sigma} = 0\,, \tag{8}$$

where ρ denotes the mass density, $\boldsymbol{u}$ and $\boldsymbol{\sigma}$ represent the corresponding displacement and stress fields, respectively. The computational domain is enclosed by a unit cell $[0,1]^2$ of which the center exists a randomly shaped void, as described by its radius $r = 0.2 + \frac{0.2}{1+exp(\tilde{r})}$ and $\tilde{r} \sim \mathbb{N}(0, 4^2(-\nabla + 3^2)^{-1})$. The bottom edge is fixed and the top edge is subjected to a tensile traction of $\boldsymbol{t} = [0, 100]$. The underlying hyperelastic material is of the incompressible Rivlin-Saunders type. For training, we directly adopt the dataset in Li et al. (2022a), where a total of 1000, 200, 200 samples are selected for training, validation and testing, respectively. For this problem, the input is represented as point clouds and the target is the resulting stress field.

Table 1: The total number of parameters (in millions) of selected models for hyperelasticity dataset.

model	eDAFNO	iDAFNO	FNO	IFNO	Geo-FNO	GNO	DeepONet	UNet	F-FNO
nparam	2.37	0.60	2.37	0.60	3.02	2.64	3.10	3.03	3.21

Table 2: Test errors for the hyperelasticity problem, where bold numbers highlight the best method.

Model		# of training samples		
		10	100	1000
Proposed model	eDAFNO	**16.446%±0.472%**	4.247%±0.066%	**1.094%±0.012%**
	iDAFNO	16.669%±0.523%	**4.214%±0.058%**	1.207%±0.006%
Baseline model	FNO w/ mask	19.487%±0.633%	7.852%±0.130%	4.550%±0.062%
	IFNO w/ mask	19.262%±0.376%	7.700%±0.062%	4.481%±0.022%
	Geo-FNO	28.725%±2.600%	10.343%±4.446%	2.316%±0.283%
	GNO	29.305%±0.321%	18.574%±0.584%	13.007%±0.729%
	DeepONet	35.334%±0.179%	25.455%±0.245%	11.998%±0.786%
	F-FNO	35.672%±3.852%	12.135%±5.813%	3.193%±1.622%
	UNet	98.167%±0.236%	34.467%±2.858%	5.462%±0.048%
Ablation study	FNO w/ smooth χ	17.431%±0.536%	5.479%±0.186%	1.415%±0.025%
	IFNO w/ smooth χ	17.145%±0.432%	5.088%±0.146%	1.509%±0.018%

Ablation study We first carry out an ablation study by comparing (a) the proposed two DAFNO models with (b) the baseline FNO/IFNO with the sharp characteristic function as input (denoted as FNO/IFNO w/ mask), and (c) the original FNO/IFNO with our proposed smoothened boundary

characteristic function as input (denoted as FNO/IFNO w/ smooth χ). In this study, scenarios (b) and (c) aim to investigate the effects of our proposed boundary smoothing technique, and by comparing scenarios (a) and (c) we verify the effectiveness of encoding the boundary information in eDAFNO architecture. In addition, both FNO- and IFNO-based models are tested, with the purpose to evaluate the model expressivity when using layer-independent parameters in iterative layers. Three training dataset sizes (i.e., 10, 100, and 1000) are employed to explore the effect of the proposed algorithms on small, medium, and large datasets, respectively. The number of trainable parameters are reported in Table 1. Following the common practice as in Li et al. (2022a), the hyperparameter choice of each model is selected by tuning the number of layers and the width (channel dimension) keeping the total number of parameters of the same magnitude.

The results of the ablation study are listed in Table 2. Firstly, by directly comparing the results of FNO (and IFNO) with mask and with smooth boundary encoding, one can tell that the boundary smoothing technique helps to reduce the error. This is supported by the observation that FNO and IFNO with smooth χ consistently outperform their counterparts with mask in all data regimes, especially when a sufficient amount of data becomes available where a huge boost in accuracy can be achieved (by over 300%). On the other hand, by encoding the geometry information into the iterative layer, the prediction accuracy is further improved, where eDAFNO and iDAFNO outperform FNO and IFNO with smoothed χ by 22.7% and 20.0% in large-data regime, respectively. Another interesting finding is that eDAFNO is 9.4% more accurate compared to iDAFNO in the large-data regime, although only a quarter of the total number of parameters is needed in iDAFNO due to its layer-independent parameter setting. This effect is less pronounced as we reduce the amount of available data for training, where the performance of iDAFNO is similar to that of eDAFNO in the small- and medium-data regimes. This is because iDAFNO has a much smaller number of trainable parameters and therefore is less likely to overfit with small datasets. Given that the performance of eDAFNO and iDAFNO is comparable, it is our opinion that both architectures are useful in different applications. In Figure 3, an example of the computational domain, the smoothened boundary encoding, the ground truth solution, and the eDAFNO prediction are demonstrated. To demonstrate the capability of prediction across resolutions, we train eDAFNO using data with 41×41 grids then apply the model to provide prediction on 161×161 grids–one can see that eDAFNO can generalize across different resolutions.

Comparison against additional baselines We further compare the performance of DAFNO against additional relevant baselines, including GNO, Geo-FNO, F-FNO, DeepONet, and UNet. Note that the results of GNO, DeepONet, and UNet are obtained using the same settings as in Li et al. (2022a). Overall, the two DAFNO variants are significantly superior to other baselines in accuracy, with eDAFNO outperforming GNO, DeepONet, UNet, Geo-FNO, and F-FNO by 1088.9%, 975.0%, 399.3%, 111.7%, and 191.9%, respectively, in large-data regime. DAFNO is also more memory efficient compared to Geo-FNO (the most accurate baseline), as it foregoes the need for additional coordinate deformation network. As shown in Table 1, when the layer-independent parameter setting in iDAFNO is taken into account, DAFNO surpasses Geo-FNO by 407.4% in memory saving.

4.2 Airfoil design

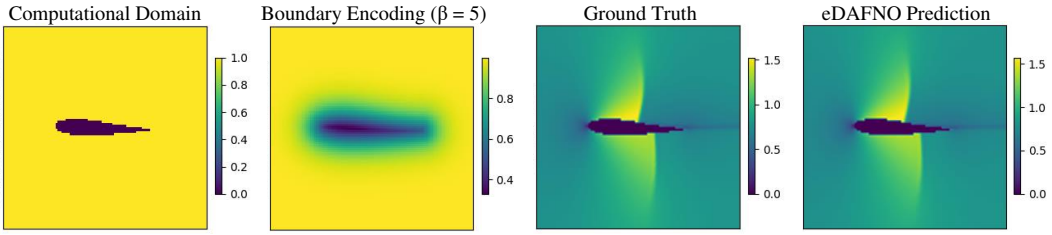

Figure 4: An illustration on a test sample from the airfoil design problem. From left to right: an illustration of the discretized computational domain, the smoothed boundary encoding (i.e., smoothed χ), the ground truth, and the eDAFNO prediction.

In this example, we investigate DAFNO's performance in learning transonic flow around various airfoil designs. Neglecting the effect of viscosity, the underlying physics can be described by the Euler equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{v}) = 0\,, \quad \frac{\partial \rho \boldsymbol{v}}{\partial t} + \nabla \cdot (\rho \boldsymbol{v} \otimes \boldsymbol{v} + p\mathbb{I}) = 0\,, \quad \frac{\partial E}{\partial t} + \nabla \cdot ((E+p)\,\boldsymbol{v}) = 0\,, \tag{9}$$

Table 3: Results for the airfoil design problem, where bold numbers highlight the best method.

	Model	Train error	Test error
Proposed model	eDAFNO	0.329%±0.020%	**0.596%±0.005%**
	iDAFNO	0.448%±0.012%	0.642%±0.020%
	eDAFNO on irregular grids	0.331%±0.003%	0.659%±0.007%
Baseline model	Geo-FNO	1.565%±0.180%	1.650%±0.175%
	F-FNO	0.566%±0.066%	0.794%±0.025%
	FNO w/ mask	2.676%±0.054%	3.725%±0.108%
	UNet w/ mask	2.781%±1.084%	4.957%±0.059%

with ρ, p and E being the fluid density, pressure and the total energy, respectively, and $\boldsymbol{v}$ denoting the corresponding velocity field. The applied boundary conditions are: $\rho_\infty = 1$, Mach number $M_\infty = 0.8$, and $p_\infty = 1$ on the far field, with no penetration enforced on the airfoil. The dataset used for training is directly taken from Li et al. (2022a), which consists of variations of the NACA-0012 airfoil and is divided into 1000, 100, 100 samples for training, validation and testing, respectively. For this problem, we aim to learn the resulting Mach number field based on a given mesh as input. An example of the computational domain, the smoothened boundary encoding, the ground truth, and the eDAFNO prediction is illustrated in Figure 4.

We report in Table 3 our experimental observations using eDAFNO and iDAFNO, along with the comparison against FNO, Geo-FNO, F-FNO and UNet, whose models are directly attained from Li et al. (2022a); Tran et al. (2022). We can see that the proposed eDAFNO achieves the lowest error on the test dataset, beating the best result of non-DAFNO baselines by 24.9% and Geo-FNO by 63.9%, respectively. Additionally, iDAFNO reaches a similar level of accuracy with only a quarter of the total number of parameters employed, which is consistent with the findings in the previous example. With the properly trained DAFNO models, efficient optimization and inverse design are made possible.

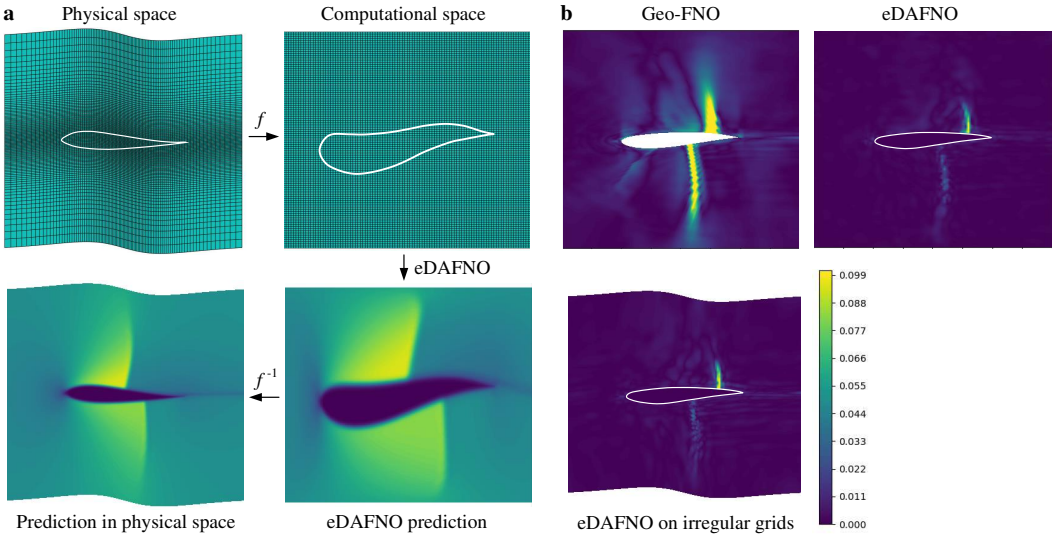

Figure 5: eDAFNO applied to the airfoil dataset on irregular grids. (a): the highly irregular and adaptive grids in the physical space is firstly deformed (via either an analytical mapping f or a trainable neural network for grid deformation) to uniform grids in the computational space, on which eDAFNO is applied to learn the underlying physics. The learned prediction is then converted back to the physical space via the inverse mapping f^{-1} or with another trainable neural network. (b): an illustration of the absolute error distribution of predictions in Geo-FNO, eDAFNO trained on uniform grids, and eDAFNO trained on irregular grids.

Aside from the enhanced predictability, DAFNO can also be easily combined with the grid mapping technique in Geo-FNO to handle non-uniform grids. In particular, no modification on the model architecture is required, where one just needs to include an analytical or trainable mapping from

non-uniform/irregular mesh grids to uniform mesh grids. As a demonstration, we consider irregular grids of the airfoil dataset and use a pre-computed function to map irregular grids to regular grids. In the irregular grid set, we place more grid points near the airfoil to provide a better resolution near the important parts. The test error is provided in Table 3, where the test loss using the eDAFNO learned model is $0.659\% \pm 0.007\%$, which is similar to the DAFNO results on uniform grids. In Figure 5, we demonstrate the irregular mesh and the absolute error comparison across Geo-FNO, DAFNO on regular grids, and DAFNO on irregular grids. One can see that, while both DAFNOs substantially outperform Geo-FNO, the error contours from eDAFNO with an irregular mesh show a smaller miss-match region near the airfoil, illustrating the flexibility of DAFNO in meshing and its capability in resolving fine-grained features.

4.3 Crack propagation with topology change in domain

In this example, we aim to showcase DAFNO's capability in handling evolving domains by modeling crack propagation in brittle fracture. We emphasize that DAFNO represents the first neural operator that allows for learning with topology change. In the field of brittle fracture, a growing crack can be viewed as a change in topology, which corresponds to an evolving $\chi(t)$ in the DAFNO architecture. In particular, we define the following time-dependent characteristic function:

$$\chi(\boldsymbol{x}, t) = \begin{cases} 1 & \boldsymbol{x} \in \Omega(t) \\ 0 & \boldsymbol{x} \in \mathbb{T} \setminus \Omega(t) \end{cases}, \tag{10}$$

where $\Omega(t)$ denotes the time-dependent domain/evolving topology. Employing time-dependent χ in (6) keeps the neural operator informed about the evolving topology. In general, the topology evolution rule that determines $\Omega(t)$ can be obtained from a separate neural network or from physics as is the case in the current example. The high-fidelity synthetic data in this example is generated using 2D PeriFast software (Jafarzadeh et al., 2022a), which employs a peridynamics (PD) theory for modeling fracture (Bobaru et al., 2016). A demonstration of the evolving topology, as well as further details regarding the governing equations and data generation strategies, is provided in Appendix B.3.

In this context, we select eDAFNO as the surrogate model to learn the the internal force density operator. Specifically, let $u_1(\boldsymbol{x}, t)$, $u_2(\boldsymbol{x}, t)$ denote the two components of the displacement field $\boldsymbol{u}$ at time t, and L_1, L_2 be the two components of the internal force density $\mathcal{L}[\boldsymbol{u}]$. Given u_1, u_2 and χ as input data, we train two separate eDAFNO models to predict L_1 and L_2, respectively. Then, we substitute the trained surrogate models in the dynamic governing equation and adopt Velocity–Verlet time integration to update $\boldsymbol{u}$ for the next time step, whereafter Ω and χ are updated accordingly.

The problem of interest is defined on a 40 mm $\times$ 40 mm thin plate with a pre-crack of length 10 mm at one edge, which is subjected to sudden, uniform, and constant tractions on the top and bottom edges. Depending on the traction magnitude, crack grows at different speeds, may or may not bifurcate, and the final crack patterns can be of various shapes. Our training data is comprised of two parts, one consisting of 450 snapshots from a crack propagation simulation with a fixed traction magnitude of $\sigma = 4$ MPa, and the other consisting of randomized sinusoidal displacement fields and the corresponding L_1, L_2 fields computed by the PD operator. The sinusoidal subset contains 4,096 instances without fracture and is used to diversify the training data and mitigate overfitting on the crack data. For testing, we use the trained models in two scenarios with traction magnitudes different from what is used in training, which allows us to evaluate the generalizability of eDAFNO. Note that these tests are highly challenging for the trained models, as the predictions of the previous time step are used as the models' input in the current step, which leads to error accumulation as the simulation marches forward.

Figure 6 displays the test results on the crack patterns under different traction magnitudes, where the low traction magnitude (left column) shows a slow straight crack, the mid-level traction results in a crack with a moderate speed and a bifurcation event (middle column), and the highest traction leads to a rapid crack growth and initiates a second crack from the other side of the plate (right column). The trained eDAFNO model is able to accurately predict the left and right examples while only seeing the middle one, indicating that it has correctly learned the true constitutive behavior of the material and generalized well to previously unseen loading scenarios and the correspondingly changed domain topology. To further verify eDAFNO's generalizability to unseen geometries, in Figure 7 we compare eDAFNO with the baseline FNO w/ smoothed mask model, and plot their relative errors in χ and $\boldsymbol{u}$. As observed, the FNO predictions become unstable at a fairly early time in the two test scenarios,

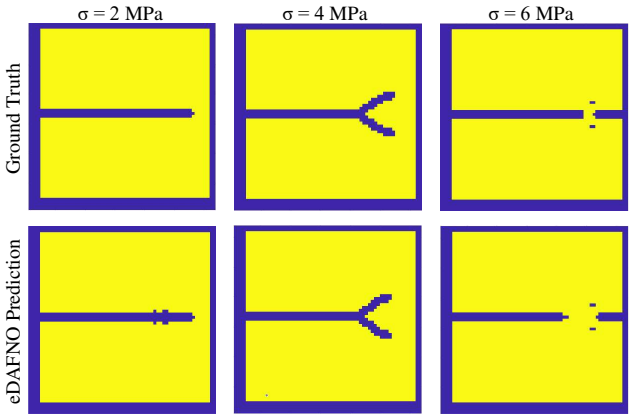

Figure 6: Comparison of the fracture patterns with different loading scenarios between the high-fidelity solution and eDAFNO prediction.

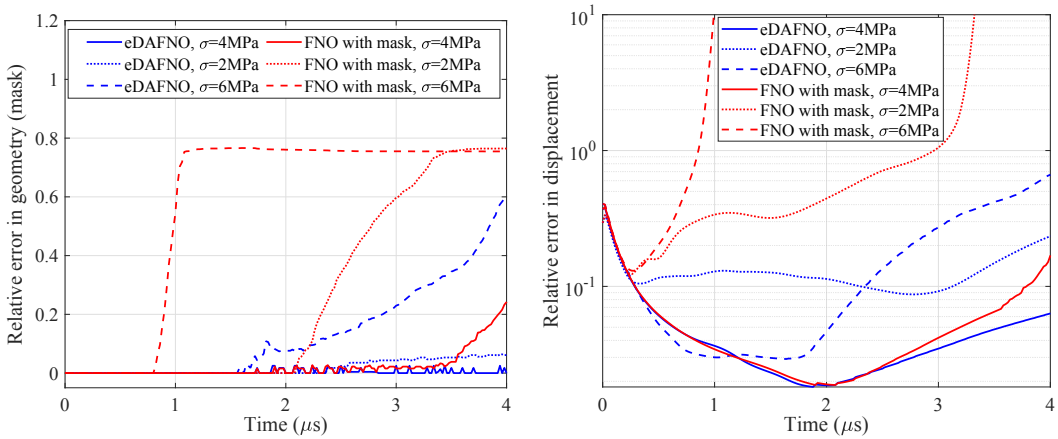

Figure 7: Comparison of relative errors in the characteristic function χ (left) and displacement fields (right) of the evolving topology predicted using the trained eDAFNO and FNO with mask at each time step for the training (loading = 4 MPa) and testing (loading = 2 MPa and 6 MPa) scenarios.

since it creates a mask which is out of training distribution, while DAFNO can handle different crack patterns by hard coding it in the architecture, so the results remain stable for a much longer time.

5 Conclusion

We introduce two DAFNO variants to enable FNOs on irregular domains for PDE solution operator learning. By incorporating the geometric information from a smoothed characteristic function in the iterative Fourier layer while retaining the convolutional form, DAFNO possesses the computational efficiency from FFT together with the flexibility to operate on different computational domains. As a result, DAFNO is not only highly efficient and flexible in solving problems involving arbitrary shapes, but it also manifests its generalizability on evolving domains with topology change. In two benchmark datasets and a real-world crack propagation dataset, we demonstrate the state-of-the-art performance of DAFNO. We find both architectures helpful in practice: eDAFNO is slightly more accurate while iDAFNO is more computationally efficient and less overfitting with limited data.

Limitation: Due to the requirement of the standard FFT package, in the current DAFNO we focus on changing domains with uniformly meshed grids. However, we point out that this limitation can be lifted by using nonuniform FFT (Greengard & Lee, 2004) or an additional mapping for grid deformation, as shown in the airfoil experiment. Additionally, in our applications, we have focused on applying the same types of boundary conditions to the changing domains (e.g., all Dirichlet or all Neumann). In this context, another limitation and possible future direction would be on the transferability to PDEs with different types of boundary conditions.

Acknowledgments and Disclosure of Funding

S. Jafarzadeh would like to acknowledge support by the AFOSR grant FA9550-22-1-0197, and Y. Yu would like to acknowledge support by the National Science Foundation under award DMS-1753031 and the AFOSR grant FA9550-22-1-0197. Portions of this research were conducted on Lehigh University's Research Computing infrastructure partially supported by NSF Award 2019035.

References

Agwai, A., Guven, I., and Madenci, E. Predicting crack propagation with peridynamics: a comparative study. *International journal of fracture*, 171(1):65, 2011.

Benson, D. J. and Okazawa, S. Contact in a multi-material Eulerian finite element formulation. *Computer Methods in Applied Mechanics and Engineering*, 193(39):4277–4298, 2004. ISSN 0045-7825.

Besnard, G., Hild, F., and Roux, S. "finite-element" displacement fields analysis from digital images: application to portevin–le châtelier bands. *Experimental mechanics*, 46(6):789–803, 2006.

Bobaru, F., Foster, J. T., Geubelle, P. H., and Silling, S. A. *Handbook of peridynamic modeling*. CRC press, 2016.

Brigham, E. O. *The fast Fourier transform and its applications*. Prentice-Hall, Inc., 1988.

Cai, S., Mao, Z., Wang, Z., Yin, M., and Karniadakis, G. E. Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica*, pp. 1–12, 2022.

Cao, S. Choose a transformer: Fourier or galerkin. *Advances in neural information processing systems*, 34:24924–24940, 2021.

Carleo, G., Cirac, I., Cranmer, K., Daudet, L., Schuld, M., Tishby, N., Vogt-Maranto, L., and Zdeborová, L. Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4): 045002, 2019.

Day, S. J., Mullineux, N., and Reed, J. Developments in obtaining transient response using fourier transforms: Part i: Gibbs phenomena and fourier integrals. *International Journal of Electrical Engineering Education*, 3(4):501–506, 1965.

De Luycker, E., Benson, D. J., Belytschko, T., Bazilevs, Y., and Hsu, M.-C. X-FEM in isogeometric analysis for linear fracture mechanics. *International Journal for Numerical Methods in Engineering*, 87:541–565, 2011.

Ghaboussi, J., Garrett Jr, J., and Wu, X. Knowledge-based modeling of material behavior with neural networks. *Journal of engineering mechanics*, 117(1):132–153, 1991.

Ghaboussi, J., Pecknold, D. A., Zhang, M., and Haj-Ali, R. M. Autoprogressive training of neural network constitutive models. *International Journal for Numerical Methods in Engineering*, 42(1): 105–126, 1998.

Goswami, S., Bora, A., Yu, Y., and Karniadakis, G. E. Physics-informed neural operators. *2022 arXiv preprint arXiv:2207.05748*, 2022.

Gottlieb, D. and Shu, C.-W. On the gibbs phenomenon and its resolution. *SIAM review*, 39(4): 644–668, 1997.

Greengard, L. and Lee, J.-Y. Accelerating the nonuniform fast fourier transform. *SIAM review*, 46(3): 443–454, 2004.

Gupta, G., Xiao, X., and Bogdan, P. Multiwavelet-based operator learning for differential equations. *Advances in Neural Information Processing Systems*, 34:24048–24062, 2021.

Ha, Y. D. and Bobaru, F. Studies of dynamic crack propagation and crack branching with peridynamics. *International Journal of Fracture*, 162(1):229–244, 2010.

Hao, Z., Wang, Z., Su, H., Ying, C., Dong, Y., Liu, S., Cheng, Z., Song, J., and Zhu, J. Gnot: A general neural operator transformer for operator learning. In *International Conference on Machine Learning*, pp. 12556–12569. PMLR, 2023.

He, J., Li, L., Xu, J., and Zheng, C. ReLu deep neural networks and linear finite elements. arXiv:1807.03973, 2018.

He, Q., Laurence, D. W., Lee, C.-H., and Chen, J.-S. Manifold learning based data-driven modeling for soft biological tissues. *Journal of Biomechanics*, 117:110124, 2021.

Jafarzadeh, S., Mousavi, F., and Bobaru, F. Perifast/dynamics: a matlab code for explicit fast convolution-based peridynamic analysis of deformation and fracture. 2022a.

Jafarzadeh, S., Mousavi, F., Larios, A., and Bobaru, F. A general and fast convolution-based method for peridynamics: Applications to elasticity and brittle fracture. *Computer Methods in Applied Mechanics and Engineering*, 392:114666, 2022b.

Kamensky, D., Hsu, M.-C., Yu, Y., Evans, J. A., Sacks, M. S., and Hughes, T. J. Immersogeometric cardiovascular fluid–structure interaction analysis with divergence-conforming b-splines. *Computer methods in applied mechanics and engineering*, 314:408–472, 2017.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Kuhl, E., Hulshoff, S., and de Borst, R. An arbitrary Lagrangian Eulerian finite element approach for fluid-structure interaction phenomena. *International Journal for Numerical Methods in Engineering*, 57:117–142, 2003.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020a.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Stuart, A., Bhattacharya, K., and Anandkumar, A. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33:NeurIPS 2020, 2020b.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier Neural Operator for Parametric Partial Differential Equations. In *International Conference on Learning Representations*, 2020c.

Li, Z., Huang, D. Z., Liu, B., and Anandkumar, A. Fourier neural operator with learned deformations for PDEs on general geometries. *arXiv preprint arXiv:2207.05209*, 2022a.

Li, Z., Meidani, K., and Farimani, A. B. Transformer for partial differential equations' operator learning. *arXiv preprint arXiv:2205.13671*, 2022b.

Liu, N., Yu, Y., You, H., and Tatikola, N. INO: Invariant neural operators for learning complex physical systems with momentum conservation. *arXiv preprint arXiv:2212.14365*, 2022.

Lu, L., Jin, P., and Karniadakis, G. E. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

Lu, L., He, H., Kasimbeg, P., Ranade, R., and Pathak, J. One-shot learning for solution operators of partial differential equations. *arXiv preprint arXiv:2104.05512*, 2021a.

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, 2021b.

Lu, L., Meng, X., Cai, S., Mao, Z., Goswami, S., Zhang, Z., and Karniadakis, G. E. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022.

Ong, Y. Z., Shen, Z., and Yang, H. IAE-NET: Integral autoencoders for discretization-invariant learning. 03 2022. doi: 10.13140/RG.2.2.25120.87047/2.

Pfau, D., Spencer, J. S., Matthews, A. G., and Foulkes, W. M. C. Ab initio solution of the many-electron schrödinger equation with deep neural networks. *Physical Review Research*, 2(3):033429, 2020.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Shadden, S. C., Astorino, M., and Gerbeau, J.-F. Computational analysis of an aortic valve jet with Lagrangian coherent structures. *Chaos*, 20:017512, 2010.

Silling, S. A. Dynamic fracture modeling with a meshfree peridynamic code. *Computational Fluid and Solid Mechanics*, 2003:641–644, 2003.

Simo, J. C. and Laursen, T. A. An augmented Lagrangian treatment of contact problems involving friction. *Computers & Structures*, 42(1):97–116, 1992. ISSN 0045-7949.

Tran, A., Mathews, A., Xie, L., and Ong, C. S. Factorized fourier neural operators. In *The Eleventh International Conference on Learning Representations*, 2022.

Tripura, T. and Chakraborty, S. Wavelet neural operator: a neural operator for parametric partial differential equations. *arXiv preprint arXiv:2205.02191*, 2022.

Yin, M., Ban, E., Rego, B. V., Zhang, E., Cavinato, C., Humphrey, J. D., and Em Karniadakis, G. Simulating progressive intramural damage leading to aortic dissection using DeepONet: an operator–regression neural network. *Journal of the Royal Society Interface*, 19(187):20210670, 2022a.

Yin, M., Zhang, E., Yu, Y., and Karniadakis, G. E. Interfacing finite elements with deep neural operators for fast multiscale modeling of mechanics problems. *Computer Methods in Applied Mechanics and Engineering, in press*, pp. 115027, 2022b.

Yin, Y., Kirchmeyer, M., Franceschi, J.-Y., Rakotomamonjy, A., et al. Continuous pde dynamics forecasting with implicit neural representations. In *The Eleventh International Conference on Learning Representations*, 2022c.

You, H., Yu, Y., D'Elia, M., Gao, T., and Silling, S. Nonlocal kernel network (NKN): A stable and resolution-independent deep neural network. *Journal of Computational Physics*, pp. arXiv preprint arXiv:2201.02217, 2022a.

You, H., Zhang, Q., Ross, C. J., Lee, C.-H., Hsu, M.-C., and Yu, Y. A physics-guided neural operator learning approach to model biological tissues from digital image correlation measurements. *Journal of Biomechanical Engineering, accepted for publication*, pp. arXiv preprint arXiv:2204.00205, 2022b.

You, H., Zhang, Q., Ross, C. J., Lee, C.-H., and Yu, Y. Learning deep implicit fourier neural operators (IFNOs) with applications to heterogeneous material modeling. *Computer Methods in Applied Mechanics and Engineering*, 398:115296, 2022c. doi: https://doi.org/10.1016/j.cma.2022.115296.

Zhang, L., Han, J., Wang, H., Car, R., and Weinan, E. Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics. *Physical Review Letters*, 120(14):143001, 2018.

A Detailed iDAFNO architecture

Similar to the eDAFNO architecture shown in (6), we present the iDAFNO version by incorporating the layer-independent parameter definition characterized in the IFNO structure (You et al., 2022c):

$$\begin{aligned} \mathcal{J}[\boldsymbol{h}](\boldsymbol{x}) :=& \boldsymbol{h}(\boldsymbol{x}) + \tau\sigma\bigg(\chi(\boldsymbol{x})\big(\mathcal{I}(\chi(\cdot)\boldsymbol{h}(\cdot);\boldsymbol{v}) - \boldsymbol{h}(\boldsymbol{x})\mathcal{I}(\chi(\cdot);\boldsymbol{v}) + W\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{c}\big)\bigg)\,, \\ &\text{where}\quad \mathcal{I}(\circ;\boldsymbol{v}) := \mathcal{F}^{-1}\big[\mathcal{F}[\kappa(\cdot;\boldsymbol{v})]\cdot\mathcal{F}[\circ]\big]\,. \end{aligned} \tag{11}$$

Here, $\tau = \dfrac{1}{L}$ is the reciprocal of the total number of layers employed. Note that the superscript l is dropped because the model parameters are layer-independent in the iDAFNO architecture, which leads to significant computational saving.

B Problem settings and additional experimental results

In order to maintain consistency with other baselines, the dimension of representation in the first two examples is set to $d_h = 32$, with a total of 4 Fourier layers and 12 Fourier modes being used in each direction. The output at each point is obtained via a projection layer in the form of a 2-layer multilayer perceptron (MLP) with width $(d_h, 128, d_u)$, where d_u is the intended number of output. In the third example, d_h is set to 16. A total of 3 Fourier layers with 32 Fourier modes in each direction are employed. The width of the projection MLP is set to $(d_h, 2d_h, d_u)$.

B.1 Experiment 1 – Constitutive modeling of hyperelasticity

The dataset is obtained from Li et al. (2022a), which consists of an interpolated dataset of 41×41 point cloud on uniformly structured grids. The parameter of each method is given in the following, where the parameter choice of each model is selected by tuning the number of layers and the width (channel dimension) keeping the total number of parameters on the same magnitude.

- eDAFNO: In these cases, we use neural operators to construct mapping from grid location $\boldsymbol{x}$ as the input, and the stress field as the output. To perform fair comparison with the results reported in Li et al. (2022a), we employ the same hyperparameters here: in particular, four Fourier layers with mode 12 and width 32 are used.
- iDAFNO: The iDAFNO cases employ the same hyperparameters as the eDAFNO cases, with the iterative layer structure demonstrated in (11). In iDAFNO, all Fourier layers share the same set of trainable parameters, while different layers have different parameters in eDAFNO. Hence, iDAFNO reduces the number of trainable parameters by almost 75%, when using the same hyperparameters as in eDAFNO.
- FNO (with mask or smooth χ): Following the same practice as in Li et al. (2022a), we train a plain FNO model (Li et al., 2020c), with the input as $[\boldsymbol{x}, \chi(\boldsymbol{x})]$ (in the "with mask" cases) or as $[\boldsymbol{x}, \tilde{\chi}(\boldsymbol{x})]$ (in the "with smoothed χ" cases). Herein, we employ the same FNO architecture as reported in Li et al. (2022a), where four Fourier layers are used with mode 12 and width 32.
- IFNO (with mask or smooth χ): Similar to the FNO cases, we also use $[\boldsymbol{x}, \chi(\boldsymbol{x})]$ (in the "with mask" cases) or $[\boldsymbol{x}, \tilde{\chi}(\boldsymbol{x})]$ (in the "with smoothed χ" cases) as the input, with four Fourier layers, mode 12, and width 32. On the Fourier layers, the implicit architecture proposed in You et al. (2022c) is employed, such that all four Fourier layers share the same set of trainable parameters. Therefore, the number of trainable parameters in IFNOs is roughly 1/4 of that in FNOs.
- Geo-FNO: As a baseline model for FNOs with various geometries, we employ the Geo-FNO architecture from Li et al. (2022a), where an additional deformation neural network is trained together with FNO to provide a diffeomorphism from uniform grids to the deformed domain.
- F-FNO: Following the settings in Li et al. (2022a), we train the F-FNO model (Li et al., 2020c) with the input $[\boldsymbol{x}, \chi(\boldsymbol{x})]$. We adopt the same F-FNO architecture as reported in Tran et al. (2022), where four Fourier layers are used with mode 16 and width 64.

- GNO: The graph neural operators are flexible on the problem geometry, which have been widely used for complex geometries (Li et al., 2020a; Liu et al., 2022). To carry out fair comparison, we build a full graph with edge connection radius $r = 0.2$, width 32 and kernel width 512. As a result, the total number of parameters in GNOs is on the same magnitude as in FNOs.
- DeepONet: As another neural operator baseline model, the deep operator network (Lu et al., 2022) is composed of two neural networks – a trunk net and a branch net to represent the basis and coefficients of the operator. In this baseline, we use five layers for both the trunk net and branch net, each with a width of 256.
- UNet: Analogous to the setup in Li et al. (2022a), we train a UNet model (Ronneberger et al., 2015) on uniform grids, where 4 downsampling and upsampling blocks with 20 hidden channels are employed.

The comparison of the total number of parameters of the selected models used in the hyperelasticity problem is listed in Table 1[3]. In addition, the average runtime for each method on the hyperelasticity problem with 1000 training samples is provided in Table 4. All tests are performed on a NVIDIA RTX A6000 GPU card with 48GB memory. From this table, we can see that in DAFNOs the runtime increases slightly compared with the corresponding FNOs, but they are still substantially more efficient than other baselines, such as Geo-FNO.

For each method, we tune the learning rate from the range [1e-3,1e-1], the decay rate from the range [0.4,0.9], the weight decay parameter from from the range [1e-6,1e-2], and the smoothing coefficient (where applicable) from the range [5,100], then report the model with the best validation error. A typical training curve can be found in Figure 8. As a supplement of Table 2, the full table of all training and testing errors from different models is provided in Table 5.

Table 4: The per-epoch runtime (in seconds) of selected models for the hyperelasticity problem.

model	eDAFNO	iDAFNO	FNO	IFNO	Geo-FNO	GNO	DeepONet	UNet	F-FNO
runtime	2.00	1.70	1.81	1.62	5.12	98.37	940.12	5.04	3.41

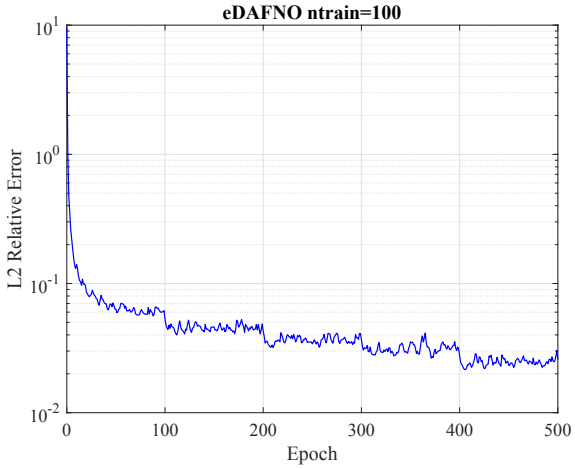

Figure 8: Demonstration of a typical training curve for eDAFNO.

To demonstrate the effect of the smoothing level when using different smoothing coefficient β, we illustrate the smoothed χ on an exemplar test sample in Figure 9. We also perform tests on the hyperelasticity example with a total of 1000 training samples and show the errors on the test dataset in Table 6. For each value of β, we search for the optimal initial learning rate, the decay rate, and the weight decay parameter based on the validation dataset, and report the optimal values.

[3]We note that the numbers of trainable parameters for the "Geo-FNO" and "FNO" cases are different from the ones provided in Li et al. (2022a). For fair comparison with methods using real-valued trainable parameters, we count each complex-valued trainable parameter as two degrees of freedom.

Table 5: Results for the hyperelasticity problem, where bold numbers highlight the best method according to the test error.

Model, Dataset	# of training samples		
	10	100	1000
eDAFNO, train	6.800%±0.670%	2.050%±0.035%	0.664%±0.014%
eDAFNO, test	**16.446%±0.472%**	4.247%±0.066%	**1.094%±0.012%**
iDAFNO, train	7.266%±0.923%	2.038%±0.036%	0.812%±0.012%
iDAFNO, test	16.669%±0.523%	**4.214%±0.058%**	1.207%±0.006%
FNO w/ mask, train	2.907%±0.318%	2.277%±0.240%	0.881%±0.015%
FNO w/ mask, test	19.487%±0.633%	7.852%±0.130%	4.550%±0.062%
FNO w/ smooth χ, train	2.876%±0.152%	2.058%±0.132%	0.815%±0.012%
FNO w/ smooth χ, test	17.431%±0.536%	5.479%±0.186%	1.415%±0.025%
Geo-FNO, train	0.547%±0.336%	0.689%±0.676%	1.192%±0.232%
Geo-FNO, test	28.725%±2.600%	10.343%±4.446%	2.316%±0.283%
IFNO w/ mask, train	2.274%±0.248%	1.687%±0.047%	2.701%±0.041%
IFNO w/ mask, test	19.262%±0.376%	7.700%±0.062%	4.481%±0.022%
IFNO w/ smooth χ, train	3.704%±0.299%	1.683%±0.029%	1.013%±0.014%
IFNO w/ smooth χ, test	17.145%±0.432%	5.088%±0.146%	1.509%±0.018%
GNO, train	27.337%±0.501%	18.713%±0.669%	13.321%±0.681%
GNO, test	29.305%±0.321%	18.574%±0.584%	13.007%±0.729%
DeepONet, train	23.071%±5.963%	22.700%±0.984%	7.937%±0.309%
DeepONet, test	35.409%±0.408%	25.925%±0.724%	11.760%±0.827%
UNet, train	98.042%±0.260%	34.569%±2.676%	1.760%±0.115%
UNet, test	98.167%±0.236%	34.467%±2.858%	5.462%±0.048%

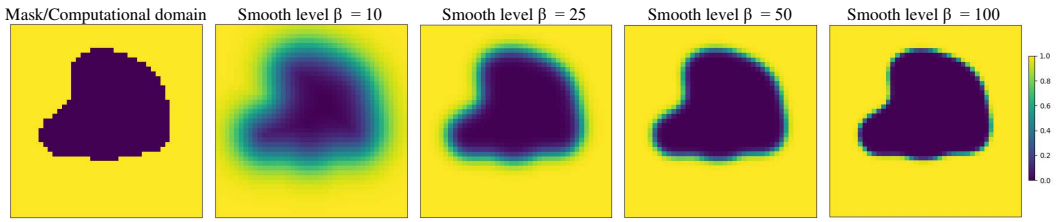

Figure 9: An illustration of the effect of varying the smoothing coefficient on the resulting boundary encoding. The larger the smoothing level β is, the sharper and narrower the encoded boundary becomes. In effect, β can be treated as a hyperparameter and tuned according to the validation error to either smoothen the boundary or keep the original boundary untouched.

Table 6: The effect of the smoothing coefficient β on test loss in the hyperelasticity example with a total of 1000 training samples.

initial learning rate	decay rate	weight decay parameter	β	train loss	test loss
4.5×10^{-2}	0.5	3×10^{-6}	5	0.564%	1.155%
4.0×10^{-2}	0.5	1×10^{-5}	10	0.637%	1.064%
2.0×10^{-2}	0.5	1×10^{-5}	20	0.454%	1.120%
1.5×10^{-2}	0.5	3×10^{-5}	30	0.516%	1.147%
2.5×10^{-2}	0.5	3×10^{-5}	40	0.608%	1.135%
1.5×10^{-2}	0.5	3×10^{-5}	50	0.504%	1.179%
1.5×10^{-2}	0.5	2×10^{-5}	60	0.498%	1.194%
1.5×10^{-2}	0.5	3×10^{-5}	70	0.508%	1.240%
1.5×10^{-2}	0.5	3×10^{-5}	80	0.515%	1.275%
1.5×10^{-2}	0.5	3×10^{-5}	90	0.529%	1.306%
3.0×10^{-2}	0.5	3×10^{-5}	100	0.675%	1.338%

B.2 Experiment 2 – Airfoil design

The airfoil dataset is directly taken from Li et al. (2022a), which is an interpolated dataset of 101×101 point cloud on uniformly structured grids. The analytical mapping function f and the corresponding inverse mapping function f^{-1} used in Figure 5 are defined in the following:

$$\begin{bmatrix} x \\ y \end{bmatrix} = f\left(\begin{bmatrix} X \\ Y \end{bmatrix}\right) = \begin{bmatrix} 0.909 \tan^{-1}(1.965X) \\ 0.714 \tan^{-1}\left(3.46Y + 0.173 \sin\left(0.909\pi \tan^{-1}(1.965X)\right)\right) \end{bmatrix}, \tag{12}$$

$$\begin{bmatrix} X \\ Y \end{bmatrix} = f^{-1}\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} 0.509 \tan(1.1x) \\ 0.289 \tan(1.4y) - 0.05 \sin(\pi x) \end{bmatrix}, \tag{13}$$

where upper- and lower-case letters indicate the coordinate systems in the physical and computational spaces, respectively.

B.3 Experiment 3 – Crack propagation with topology change

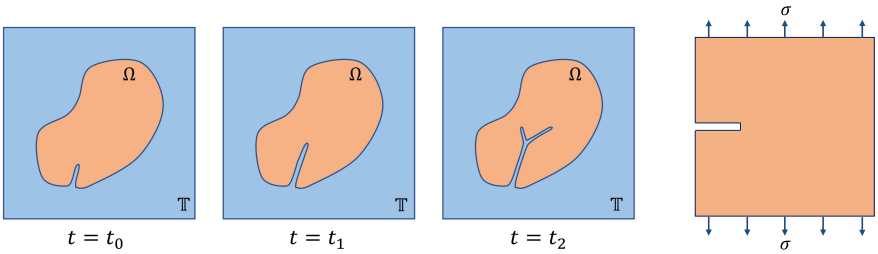

Figure 10: An illustration of the crack propagation problem, showing the topology change in DAFNO (left) and the physical problem setup (right), where a 2D plate with a pre-crack is subjected to external tractions (denoted as σ with a slight abuse of notation) on the top and bottom edges.

An illustration of the time-dependent domain evolution, as well as the problem setup, is shown in Figure 10. The governing PD equation of motion for brittle fracture used in generating the dataset is given below:

$$\rho \frac{\partial^2 \boldsymbol{u}}{\partial t^2} = \mathcal{L}(\boldsymbol{u}) + \boldsymbol{b}\,, \mathcal{L}(\boldsymbol{u})(\boldsymbol{x}, t) = \int_{\mathcal{H}_x} \mu(\boldsymbol{x}, \boldsymbol{y}, t) f(\boldsymbol{x}, \boldsymbol{y}, t) d\boldsymbol{y}\,. \tag{14}$$

In (14), $\boldsymbol{u}$ is the displacement field, ρ is mass density, t is time, $\boldsymbol{b}$ is the external force density (a.k.a. the body force), and $\mathcal{L}(\boldsymbol{u})$ is the internal force density. $\mathcal{L}(\boldsymbol{u})$ is the divergence of stress in local theory, but in PD, it is defined by the integral described in (14). $\mathcal{H}_x$ denotes a finite-size neighborhood of point $\boldsymbol{x}$. $f(\boldsymbol{x}, \boldsymbol{y}, t)$ is the dual force density representing the pairwise force acting between unit volumes at points $\boldsymbol{x}$ and $\boldsymbol{y}$ in its neighborhood $\mathcal{H}_x$. f depends on the PD constitutive model, and $\mu(\boldsymbol{x}, \boldsymbol{y}, t)$ is a binary history-dependent quantity representing material damage. μ is either 0 or 1 for brittle fracture models, where $\mu = 0$ denotes a lost interaction for material points $\boldsymbol{x}$ and $\boldsymbol{y}$ while $\mu = 1$ implies an intact connection between the two. For the material model, we choose to work with the linearized bond-based model, and for the damage model we adopt the pointwise energy-based model provided in the PeriFast software. According to the employed damage model, topology evolution due to growing crack is a function of strain energy which depends on the updated displacement field, i.e., $\Omega(t) = \Omega(\boldsymbol{u}(t))$. Additional details regarding the PD formulation can be found in Jafarzadeh et al. (2022b).

The physical parameters used in generating the data are: Young's modulus $E = 150$ GPa, Poison's ratio $\nu = 0.33$, mass density $\rho = 1000$ kg/m^3, and fracture energy $G_0 = 200$ J/m^2. The relative computational parameters are: PD horizon (the radius of the neighborhood for nonlocal interactions) $\delta = 2.07$ mm, extended domain (i.e., the periodic box) size 44.14 mm $\times$ 44.14 mm with 64×64 discretization, and time step $\Delta t = 2 \times 10^{-8}$ s. For the crack data subset in training, we run PeriFast software with the above parameters and the traction magnitude of 4 MPa. We record u_1, u_2, χ, L_1, and L_2 for 450 consecutive time steps. For the sinusoidal data subset used in training, we set $u_1 = c \sin\left(\frac{2m\pi x_1}{L}\right) \sin\left(\frac{2n\pi x_2}{L}\right)$, and $u_2 = 0$ for $m, n = 1, 2, \cdots, 32$, where L is the length of the square box, x_1 and x_2 are the 2D coordinates, and $c = 0.01/32$ is a scaling factor to make

the generated displacement in the same scale as the crack data. Additionally, we set $u_1 = 0$ and $u_2 = c \sin\left(\frac{2m\pi x_1}{L}\right) \sin\left(\frac{2n\pi x_2}{L}\right)$. This results in a total of 2,048 instances of sinusoidal displacement fields. Next, we set $\chi = 1$ for all nodes and use the PD operator in PeriFast to compute the corresponding L_1 and L_2 fields.

Following the common practice in PD simulations (Ha & Bobaru, 2010), we employ the following two additional techniques to help with training and stabilizing crack propagation. Firstly, we do not allow damage to initiate from boundaries. This technique has been used in previous PD simulations and is referred to as the "no-fail zone". It effectively stops unrealistic distributed damage from initiating on the boundaries. Secondly, given that the physical problem is symmetric, we enforce the damage growth in the simulation with eDAFNO to be symmetric as well. Note that the whole domain is used for training, and the predictions on the entire domain is used for next time step evaluation. Symmetry is enforced only when the topology characteristic function χ is updated.

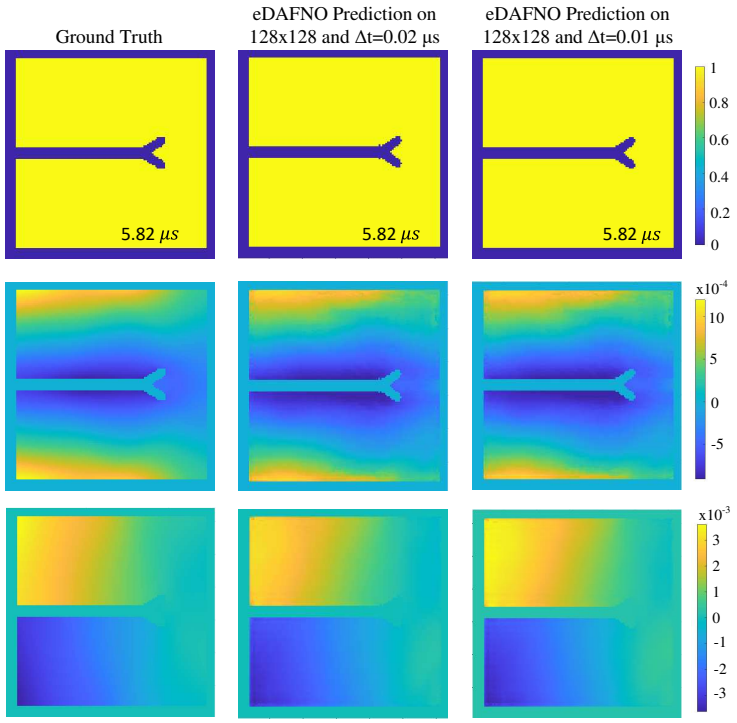

Figure 11: Demonstration of the resolution-independence property of eDAFNO trained using a spatial discretization of 64×64 and time step of 0.02 μs and tested on a spatial discretization of 128×128 and time step of 0.01 μs. The three rows correspond to the χ, u_1, and u_2 fields, respectively.

Besides the resolution-independence property of DAFNO as shown in Figure 3, we further investigate the generalizability of DAFNO in both physical and temporal resolutions with this example. Specifically, the eDAFNO model is trained on a spatial resolution of 64×64 and a time step of 0.02 μs, and it is here tested on both a finer spatial resolution of 128×128 and a finer time step of 0.01 μs. As shown in Figure 11, the performance of the low-resolution-trained eDAFNO on high resolutions is compared with the high-fidelity peridynamics simulation results, where visually identical results are observed. Note that, although the time marching is computed with an ODE solver in this example, the temporal resolution independence is still worth investigating because, as the number of time steps increases, the number of times that the error accumulates in the dynamic solver increases as well. Our results show that eDAFNO prediction remains independent of the time step employed.

B.4 Experiment 4 – Pipe flow

We perform an additional experiment of pipe flow, in which the dataset is obtained from Li et al. (2022a). We closely follow their problem setup and briefly document the comparison against Geo-FNO in what follows.

Using 1000 samples for training, eDAFNO has achieved a similar performance to Geo-FNO: when comparing the relative L^2 errors, eDAFNO's test error on the pipe dataset is 0.71%, while the test error of Geo-FNO is 0.67%. When comparing the maximum absolute error (cf. Figure 12), eDAFNO has 0.051, while Geo-FNO has 0.061. This is probably due to the fact that all pipes have a very simple geometry, which can be accurately represented with the pre-specified mapping in Geo-FNO. Note that such a pre-specified mapping for grid deformation can be easily added in DAFNO, as demonstrated in the airfoil experiment. In this circumstance, DAFNO becomes exactly the same as Geo-FNO in the pipe flow setup.

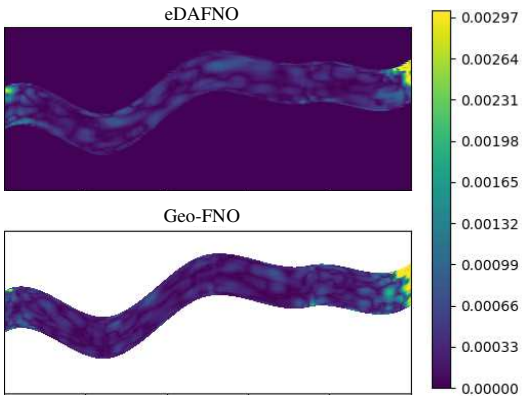

Figure 12: An illustration of the absolute error distribution of predictions from Geo-FNO and eDAFNO on the pipe dataset. The maximum absolute errors of Geo-FNO and eDAFNO are 0.061 and 0.051, respectively. In the vicinity of the outlet where most errors accumulate, eDAFNO is also more accurate compared to Geo-FNO.