Learning Deep Implicit Fourier Neural Operators (IFNOs) with Applications to Heterogeneous Material Modeling

Huaiqian You[a], Quinn Zhang[a], Colton J. Ross[b], Chung-Hao Lee[b], Yue Yu[a,*]

[a]*Department of Mathematics, Lehigh University, Bethlehem, PA 18015, USA*
[b]*School of Aerospace and Mechanical Engineering, The University of Oklahoma, Norman, OK 73019, USA*

Abstract

Constitutive modeling based on continuum mechanics theory has been a classical approach for modeling the mechanical responses of materials. However, when constitutive laws are unknown or when defects and/or high degrees of heterogeneity are present, these classical models may become inaccurate. In this work, we propose to use data-driven modeling, which directly utilizes high-fidelity simulation and/or experimental measurements to predict a material's response without using conventional constitutive models. Specifically, the material response is modeled by learning the implicit mappings between loading conditions and the resultant displacement and/or damage fields, with the neural network serving as a surrogate for a solution operator. To model the complex responses due to material heterogeneity and defects, we develop a novel deep neural operator architecture, which we coin as the Implicit Fourier Neural Operator (IFNO). In the IFNO, the increment between layers is modeled as an integral operator to capture the long-range dependencies in the feature space. As the network gets deeper, the limit of IFNO becomes a fixed point equation that yields an implicit neural operator and naturally mimics the displacement/damage fields solving procedure in material modeling problems. To obtain an efficient implementation, we parameterize the integral kernel of this integral operator directly in the Fourier space and interpret the network as discretized integral (nonlocal) differential equations, which consequently allow for the fast Fourier transformation (FFT) and accelerated learning techniques for deep networks. We demonstrate the performance of our proposed method for a number of examples, including hyperelastic, anisotropic and brittle materials. As an application, we further employ the proposed approach to learn the material models directly from digital image correlation (DIC) tracking measurements, and show that the learned solution operators substantially outperform the conventional constitutive models in predicting displacement fields.

Keywords: Operator-Regression Neural Networks, Fourier Neural Operator (FNO), Data-Driven Material Modeling, Deep Learning, Brittle Fracture, Implicit Networks

*Corresponding author
Email addresses: huy316@lehigh.edu (Huaiqian You), quz222@lehigh.edu (Quinn Zhang), cjross@ou.edu (Colton J. Ross), ch.lee@ou.edu (Chung-Hao Lee), yuy214@lehigh.edu (Yue Yu)

Contents

1. Introduction

In science and engineering, predicting and monitoring heterogeneous material responses are ubiquitous in many applications [1–11]. In these materials, the microstructure, in terms of the geometric distribution of phases, constituent properties, and interfacial bonding attributes influences the deformation and failure behavior, which needs to be accurately captured to guarantee reliable and trustworthy predictions and inform decision making. Conventionally, constitutive models based on continuum mechanics have been commonly employed for modeling heterogeneous material responses. When the material microstructures are known, constitutive models in conjunction with other field equations (e.g., balance of linear momentum) are often built in the form of partial differential equations (PDEs), and the material responses are obtained by approximating the PDE solutions with classical numerical methods such as finite elements.

However, fundamental challenges are still present in utilizing the constitutive models and numerical simulations to provide a comprehensive physical and functional description of heterogeneous material responses [12]. First, in the constitutive modeling theory the choice of governing laws (such as the strain energy density

function) is often determined *a priori* and the free parameters are often tuned to obtain agreement with experimental stress-strain data. This fact makes the rigorous calibration and validation process challenging. Second, although new experimental technologies and testing procedures have been designed to observe much smaller microstructure patterns and monitor defects in a faster manner [4, 13–18], it remains difficult to fully quantify the microstructure and responses for individual material samples, due to variability and measurement noises from different microstructure geometries, properties, and operating environments. In addition to these challenges, many microstructure characterization methods require the use of destructive methods that could alter the observed microstructural properties, such as optical clearing and histological processing [19, 20]. Therefore, in the application scenarios where the material response of a particular material sample is of interests, such as the non-destructive evaluation and damage prediction problems, conventional constitutive models may suffer from errors stemmed from its functional form assumption and the measurement noises, leading to limited predictivity.

To address these challenges, data-driven computing has been considered as an alternative to the conventional constitutive modeling. In recent years, there has been significant progress in the development of deep neural networks (NNs), focusing on learning the hidden physics of a complex system [21–35]. Among these works, several studies that use neural networks in modeling heterogeneous materials have been conducted [36–41]. In [37, 38], physics-informed NN models [42] were developed, where the material responses were modeled as the solution of a *known* PDE by a deep NN with weights and biases learned together with the PDE's unknown parameter fields (e.g., permeability). In [41], a symbolic regression method [43–46] was developed to learn the microstructure-dependent plasticity from data, where the constitutive models were generated using interpretable machine learning as symbolic expressions. In [36], data-driven approaches were employed for the homogenization procedure, where the information from multiple sub-scales can be used to sequentially generate the macroscopic prediction in a cost-efficient manner. In [39, 40], representative volume elements (RVE) were employed to build the material law for heterogeneous materials, and a homogenized model was then discovered based on the RVE database. To the authors' best knowledge, most of the state-of-the-art NN developments for heterogeneous material modeling either focus on the homogenized behavior of the material or rely on (partially) known physics laws, which limits their applicability to problems where the unknown heterogeneous behavior of each individual sample is of interest.

More recently, the use of NNs has been extended to learning maps between inputs of a dynamical system and its state, so that the network serves as a surrogate for a solution operator [47–51]. This approach, which can be referred as *neural operators*, finds applicability when the constitutive laws are unknown. Representative works in this direction include the integral neural operator architectures [49–54] and the DeepONet architectures [47, 48, 55]. Comparing with the classical NNs, the most notable advantages of neural operators are resolution independence and generalizability to different input instances. The former implies that the accuracy of the prediction is invariant with respect to the resolution of input parameters such as loading conditions and material properties. This fact is in stark contrast with the classical finite-

dimensional approaches that build the NN models between finite-dimensional Euclidean spaces, so that their accuracy is tied to the resolution of input [56–60]. Furthermore, being generalizable with respect to different input parameter instances renders another computing advantage: once the neural operator is trained, solving for a new instance of the input parameter only requires a forward pass of the network. This unique property is in contrast with traditional PDE-constrained optimization techniques [61] and some other NN models that directly parameterize the solution [42, 62–65], as all these methods only approximate the solution for a *single instance of the input*. In [55, 66, 67], neural operators have been successfully applied to model the unknown physics law of homogeneous materials. In [49–51, 68], neural operators are employed as a solution surrogate for the Darcy's flow in a heterogeneous porous medium, when the microstructure field is known.

In this work, we propose to advance the current data-driven methods on heterogeneous material modeling by designing deep neural operators to model heterogeneous material responses without using any predefined constitutive models or microstructure measurements. Specifically, through learning the solution operator directly from high-fidelity simulation and/or experimental measurements, we integrate material identification, modeling procedures, and material response prediction. The material microstructure properties are learned implicitly from the data and naturally embedded in the network parameters. The heterogeneous material responses can thus be obtained without assumptions on microstructure or governing laws. To capture the complex and possibly nonlinear material responses, deep NNs are necessary to learn multiple levels of abstraction for representations of the raw input data. To achieve this goal, we pursue a new integral neural operator architecture that, 1) is stable in the limit of deep layers with fixed memory costs, 2) has guaranteed universal approximation capability, and 3) is independent of the input resolution and generalizable to unseen input function instances. Our proposed architecture can be interpreted as a data-driven surrogate of the fixed point procedure, in the sense that the increment of fixed point iterations are modeled as increment between layers. As such, a forward pass through a very deep network is analogous to obtaining the PDE solution as an implicit problem, and the universal approximation capability is guaranteed as far as there exists a convergent fixed point equation[1]. To further accelerate the learning, we identify iterative layers with time instants such that the proposed network can be interpreted as discretized autonomous integral (non-local) differential equations, and consequently allows for the shallow-to-deep initialization technique [52, 74, 75] where optimal parameters learned on shallow networks are considered as (quasi-optimal) initial guesses for deeper networks. Since the proposed architecture is built as a modification of the Fourier Neural Operator method (FNO), it also parameterizes the integral kernel directly in the Fourier space and utilizes the fast Fourier transformation (FFT) to efficiently evaluate the integral operator. As such, our network inherits the advantages of FNOs on *resolution independence* and *superior efficiency*. Because it preserves the similar properties to both the implicit neural networks and the FNOs, we refer to our proposed network as implicit Fourier neural operators (IFNOs).

[1]Here, we point out that the idea of using constant parameters across layers and formulating the NNs as a fixed point equation was also proposed in implicit networks [69–73] such that the deep network can be trained with fixed memory costs.

We summarize our major contributions as follows.

1. We introduce a novel deep neural operator by parameterizing the layer increment as an integral operator, referred to as IFNO, which learns the mapping between loading conditions and material responses as a solution operator while preserving the accuracy across resolutions.

2. By resembling the network architecture as a fixed point method, the IFNOs can be interpreted as a numerical solver for an implicit problem with unknown material properties/microstructure, and the universal approximation property is guaranteed as far as there exists a converging fixed point equation for this implicit problem.

3. By identifying the layers with time instants, the IFNOs can also be interpreted as discretized nonlocal time-dependent equations, which allows for accelerated learning techniques for deep networks, such as the shallow-to-deep technique [74].

4. In a variety of complex material response learning tasks, the IFNOs demonstrate not only stability but also improved accuracy in the deep network limit: in complex learning tasks, the IFNOs outperform the best FNOs with reduced memory costs and halved prediction errors.

5. Our proposed method integrates material identification, modeling procedures, and material response prediction into one learning framework, which makes it particularly promising for learning complex material responses without explicit constitutive models and/or microstructure measurements. To demonstrate this capability, we learn the mechanical responses of a latex glove sample directly from digital image correlation (DIC) tracking measurements. Comparing with the conventional constitutive models, our method reduces the prediction error by 10 times.

The remainder of this paper is organized as follows. In Section 2, we introduce three integral neural operator architectures that inspired our work and highlight their advantages and limitations. In Section 3, we introduce the IFNOs as inspired by an implicit problem solver, and discuss its universal approximation capability. In Section 4, we show the stability and convergence of the IFNOs for a number of benchmarks, including heterogeneous, hyperelastic, anisotropic and brittle fracture material problems, that illustrate the efficacy of our network compared to the baseline networks. Next, in Section 5 we further demonstrate the applicability of our data-driven approach to learn the unknown mechanical responses directly from DIC tracking measurements, providing evidence that the scheme yields accurate predictions for practical engineering problems. In Section 6, we provide a summary of our achievements and concluding remarks. In the appendix, we provide additional numerical results.

2. Background and Related Work

This section provides the necessary background for the rest of the paper by formally stating the problem of neural operator learning, providing succinct reviews on the three integral neural operator learning approaches

Model	Layer-Independent Parameters	Efficiency Through FFT	Continuous in Depth (Time)	Stability in Deep Networks	Ref
GKN	✓	–	–	–	[49, 50]
NKN	✓	–	✓	✓	[52]
FNO	–	✓	–	–	[51]
IFNO	✓	✓	✓	✓	

Table 1: List of the properties for the graph kernel networks (GKNs), nonlocal kernel networks (NKNs), Fourier neural operators (FNOs), and the proposed implicit Fourier neural operators (IFNOs).

recently proposed in the literature that inspired the proposed IFNOs, and highlighting their properties, as summarized in Table 1.

2.1. Problem statement: Learning solution operators

The main application considered in this work is the modeling of complex material responses under different loading conditions. Formally, consider a s-dimensional body occupying the domain $\Omega \subset \mathbb{R}^s$ ($s = 1, 2$ or 3), which deforms under external loading. Without prior knowledge of the material properties or constitutive laws, our ultimate goal is to identify the *best surrogate solution operator*, that accurately predicts the material mechanical responses in terms of the resultant displacement field $\boldsymbol{u}(\boldsymbol{x})$ and/or damage field given new and unseen material property or loading scenarios. In this context, different types of loading scenarios are considered, such as a displacement-type loading applied on the subject's boundary, a body force applied on the whole domain Ω, a traction loading applied on part of its boundaries or a combination of the above. Denoting the whole boundaries of domain Ω as $\partial\Omega$, we consider general mixed boundary conditions: $\partial\Omega = \partial\Omega_D \bigcup \partial\Omega_N$ and $(\partial\Omega_D)^o \bigcap (\partial\Omega_N)^o = \emptyset$, where $\partial\Omega_D$ and $\partial\Omega_N$ are the Dirichlet and Neumann boundaries, respectively. To apply the displacement-type loading on the boundary, we assume that $\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{u}_D(\boldsymbol{x})$ are provided on $\partial\Omega_D$, while the traction $\boldsymbol{t}(\boldsymbol{x})$ is applied on the boundary $\partial\Omega_N$.

In this work, we propose to learn the surrogate solution operator as a mapping between functions, namely, the microstructure and/or loading and the resultant displacement/damage field, given a collection of observed function pairs. Mathematically, let $\mathcal{K}_{\boldsymbol{b}}$ be the unknown differential operator associated with the momentum balance equation and $\mathcal{N}_{\boldsymbol{b}}$ be the unknown operator associated with the traction, both depending on the material microstructure parameter field $\boldsymbol{b}(\boldsymbol{x})$. Given a body force $\boldsymbol{g}(\boldsymbol{x})$, the momentum balance equation and boundary conditions write:

$$\begin{aligned} \mathcal{K}_{\boldsymbol{b}}[\boldsymbol{u}](\boldsymbol{x}) &= \boldsymbol{g}(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \\ \boldsymbol{u}(\boldsymbol{x}) &= \boldsymbol{u}_D(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial\Omega_D, \\ \mathcal{N}_{\boldsymbol{b}}[\boldsymbol{u}](\boldsymbol{x}) &= \boldsymbol{t}(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial\Omega_N. \end{aligned} \tag{2.1}$$

To solve the displacement field, we consider the problem of learning a general solution operator, with its input being a concatenated vector function $\boldsymbol{f}(\boldsymbol{x})$ of $\boldsymbol{x}$, $\boldsymbol{b}(\boldsymbol{x})$, $\boldsymbol{g}(\boldsymbol{x})$, $\boldsymbol{u}_D(\boldsymbol{x})$, $\boldsymbol{t}(\boldsymbol{x})$ and its output being the displacement field $\boldsymbol{u}(\boldsymbol{x})$, for all $\boldsymbol{x} \in \Omega$. Here, we notice that $\boldsymbol{u}_D(\boldsymbol{x})$ and $\boldsymbol{t}(\boldsymbol{x})$ are only defined on the displacement boundary $\partial\Omega_D$ and the traction boundary $\partial\Omega_N$, respectively. To make them well-defined

on the whole domain, we employ the zero-padding strategy proposed in [68], namely, we define $\boldsymbol{f}(\boldsymbol{x}) := [\boldsymbol{x}, \boldsymbol{b}(\boldsymbol{x}), \boldsymbol{g}(\boldsymbol{x}), \tilde{\boldsymbol{u}}_D(\boldsymbol{x}), \tilde{\boldsymbol{t}}(\boldsymbol{x})]$ where

$$\tilde{\boldsymbol{u}}_D(\boldsymbol{x}) = \begin{cases} \boldsymbol{u}_D(\boldsymbol{x}), & \text{if } \boldsymbol{x} \in \partial\Omega_D \\ 0, & \text{if } \boldsymbol{x} \in \Omega \backslash \partial\Omega_D \end{cases}, \qquad \tilde{\boldsymbol{t}}(\boldsymbol{x}) = \begin{cases} \boldsymbol{t}(\boldsymbol{x}), & \text{if } \boldsymbol{x} \in \partial\Omega_N \\ 0, & \text{if } \boldsymbol{x} \in \Omega \backslash \partial\Omega_N \end{cases}. \tag{2.2}$$

In what follows, we denote the input and output function spaces as $\mathcal{F} = \mathcal{F}(\Omega; \mathbb{R}^{d_F})$ and $\mathcal{U} = \mathcal{U}(\Omega; \mathbb{R}^{d_u})$, respectively. Let $\{\boldsymbol{f}_j, \boldsymbol{u}_j\}_{j=1}^N$ be a set of observations where the input $\{\boldsymbol{f}_j\} \subset \mathcal{F}$ is a sequence of independent and identically distributed random fields from a known probability distribution μ on $\mathcal{F}$, and $\mathcal{G}^\dagger[\boldsymbol{f}_j](\boldsymbol{x}) = \boldsymbol{u}_j(\boldsymbol{x}) \in \mathcal{U}$, possibly noisy, is the output of the solution map $\mathcal{G}^\dagger : \mathcal{F} \to \mathcal{U}$. With neural operator learning, we aim to build an approximation of $\mathcal{G}^\dagger$ by constructing a nonlinear parametric map

$$\mathcal{G}[\cdot\,; \theta] : \mathcal{F} \times \Theta \to \mathcal{U},$$

in the form of a neural network (NN), for some finite-dimensional parameter space Θ. Here, $\theta \in \Theta$ is the set of parameters in the network architecture to be inferred by solving the following minimization problem

$$\min_{\theta \in \Theta} \mathbb{E}_{\boldsymbol{f} \sim \mu}[C(\mathcal{G}[\boldsymbol{f}; \theta], \mathcal{G}^\dagger[\boldsymbol{f}])] \approx \min_{\theta \in \Theta} \sum_{j=1}^{N} [C(\mathcal{G}[\boldsymbol{f}_j; \theta], \boldsymbol{u}_j)], \tag{2.3}$$

where C denotes a properly defined cost functional $C : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$. Although $\boldsymbol{f}_j$ and $\boldsymbol{u}_j$ are (vector) functions defined on a continuum, with the purpose of doing numerical simulations, we assume that they are defined on a discretization of the domain defined as $\chi = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_M\} \subset \Omega$. With such a discretization to establish learning governing laws, a popular choice of the cost functional C is the mean square error, i.e.,

$$C(\mathcal{G}[\boldsymbol{f}_j; \theta], \boldsymbol{u}_j) := \sum_{\boldsymbol{x}_i \in \chi} ||\mathcal{G}[\boldsymbol{f}_j; \theta](\boldsymbol{x}_i) - \boldsymbol{u}_j(\boldsymbol{x}_i)||^2.$$

In this context, we have formulated the material response modeling problem as to learn the solution operator $\mathcal{G}$ of an unknown PDE system from data. To emphasize the importance and challenges of learning the solution operator rather than a particular solution $\boldsymbol{u}$, we notice that when the operators $\mathcal{K}_{\boldsymbol{b}}$ and $\mathcal{N}_{\boldsymbol{b}}$ are known, existing methods, ranging from the classical discretization of PDEs with known coefficients to modern machine learning (ML) approaches such as the basic version of physics-informed neural networks [42], lead to finding the solution $\boldsymbol{u} \in \mathcal{U}$ for a single instance of the material parameter and loading $\boldsymbol{f} \in \mathcal{F}$. However, when constitutive laws are unknown or when defects and/or high degrees of heterogeneity are present such that the classical constitutive models may become inaccurate, the operators $\mathcal{K}_{\boldsymbol{b}}$ and $\mathcal{N}_{\boldsymbol{b}}$ can not be predefined.

Thus, our goal is to provide a *neural operator*, i.e., an approximated solution operator $\mathcal{G}[\cdot; \theta] : \boldsymbol{f} \to \boldsymbol{u}$ that delivers solutions of the system for any input $\boldsymbol{f}$. This is a more challenging task for several reasons. First, in contrast to the classical NN approaches where the solution operator is parameterized between finite-

dimensional Euclidean spaces [56–60], the neural operators are built as mappings between infinite-dimensional spaces, and they are resolution independent. As the consequence, *no further modification or tuning will be required for different resolutions* in order to achieve the same level of solution accuracy [49, 51, 52]. Second, for every new instance of material microstructure and/or loading scenarios $\boldsymbol{f}$, the neural operators require only a forward pass of the network, which implies that the optimization problem (2.3) *only needs to be solved once and the resulting NN can be utilized to solve for multiple instances of the input parameter.* This property is in contrast to the classical numerical PDE methods [76–78] and some ML approaches [42, 62–65], where the optimization problem needs to be solved for every new instance of the input parameter of a known governing law. Finally, of fundamental importance is the fact that the neural operators can find solution maps regardless of the presence of an underlying PDE and only require the observed data pairs $\{(\boldsymbol{f}_j, \boldsymbol{u}_j)\}_{j=1}^N$. Therefore, learning a data-driven neural operators would be particularly promising when the mechanical responses are provided by experimental measurements such as the displacement tracking data from DIC (see Section 5) or molecular dynamics simulations [79, 80] for which the material governing equations are not available.

2.2. Three relevant integral neural operator architectures

We now discuss the network architecture of three relevant integral neural operator learning methods, namely, the GKNs [49, 50], NKNs [52], and FNOs [51]. To provide a consistent description of all three networks and illustrate their connections with the proposed IFNO architecture, we describe each model following a formulation similar to the one presented in [52].

Lifting Layer. In integral neural operator models, we first lift the input $\boldsymbol{f}(\cdot) \in \mathcal{F}$ to a representation (feature) $\boldsymbol{h}(\cdot, 0)$ that corresponds to the first network layer (also known as the lifting layer, see, e.g., [81]). In this section, we identify the first argument of $\boldsymbol{h}$ with space (the set of nodes) and the second argument with time (the set of layers). Given an input vector field $\boldsymbol{f}(\boldsymbol{x}) : \mathbb{R}^s \to \mathbb{R}^{d_F}$, we define the first network layer as

$$\boldsymbol{h}(\boldsymbol{x}, 0) = \mathcal{P}[\boldsymbol{f}](\boldsymbol{x}) := P(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{p}(\boldsymbol{x}).$$

Here, $P(\boldsymbol{x}) \in \mathbb{R}^{d \times d_F}$ and $\boldsymbol{p}(\boldsymbol{x}) \in \mathbb{R}^d$ define an affine pointwise mapping. In practice, $P(\boldsymbol{x})$ and $\boldsymbol{p}(\boldsymbol{x})$ are often taken as constant parameters, i.e., $P(\boldsymbol{x}) \equiv P$ and $\boldsymbol{p}(\boldsymbol{x}) \equiv \boldsymbol{p}$.

Iterative Kernel Integration Layers. Then, we formulate the NN architecture in an iterative manner:

$$\boldsymbol{h}(\cdot, l\Delta t) = \mathcal{L}_l[\boldsymbol{h}(\cdot, (l-1)\Delta t)], \quad l = 1, \cdots, L, \tag{2.4}$$

where $\boldsymbol{h}(\cdot, j\Delta t)$, $j = 0, \cdots, L := T/\Delta t$, is a sequence of functions representing the values of the network at each hidden layer, taking values in $\mathbb{R}^d$. $\mathcal{L}_1, \cdots, \mathcal{L}_L$ are the nonlinear operator layers defined via the action of the sum of a local linear operator (i.e., a nonlocal integral kernel operator) and a bias function. Within each

layer, we treat the nodes within a layer as a continuum so that we have an infinite number of nodes, i.e., a layer has an infinite width. As such, each layer representation can be seen by a function of the continuum set of nodes $\Omega \subset \mathbb{R}^s$. Then, we denote the l-th network representation by $\boldsymbol{h}(\boldsymbol{x}, l\Delta t) : \mathbb{R}^s \times \mathbb{N}^+ \to \mathbb{R}^d$, or, equivalently, $\boldsymbol{h}(\boldsymbol{x}, l\Delta t) = \boldsymbol{h}(\boldsymbol{x}, t) : \mathbb{R}^s \times (0, T] \to \mathbb{R}^d$. Here, $l = 0$ (or equivalently, $t = 0$) denotes the first hidden layer, whereas $t = L\Delta t$ (or $t = T$) for the last hidden layer. The use of the symbol t stems from the relationship that can be established between the network update and a time stepping scheme.

Projection Layer. Third, the output $\boldsymbol{u}(\cdot) \in \mathcal{U}$ is obtained through a projection layer. In particular, we project the last hidden layer representation $\boldsymbol{h}(\cdot, T)$ onto $\mathcal{U}$ as:

$$\boldsymbol{u}(\boldsymbol{x}) = \mathcal{Q}[\boldsymbol{h}(\cdot, T)](\boldsymbol{x}) := Q_2(\boldsymbol{x})\sigma(Q_1\boldsymbol{h}(\boldsymbol{x}, T) + \boldsymbol{q}_1(\boldsymbol{x})) + \boldsymbol{q}_2(\boldsymbol{x}).$$

Here, $Q_1(\boldsymbol{x}) \in \mathbb{R}^{d_Q \times d}$, $Q_2(\boldsymbol{x}) \in \mathbb{R}^{d_u \times d_Q}$, $\boldsymbol{q}_1(\boldsymbol{x}) \in \mathbb{R}^{d_Q}$ and $\boldsymbol{q}_2(\boldsymbol{x}) \in \mathbb{R}^{d_u}$ are the appropriately sized matrices and vectors that are part of the parameter set that we aim to learn. σ is an activation function. Unless otherwise stated, in this work we choose σ to be the popular rectified linear unit (ReLU) function:

$$\mathrm{ReLU}(x) := \begin{cases} 0, & \text{for } x \leq 0; \\ x, & \text{for } x > 0. \end{cases} \tag{2.5}$$

Similarly as for the lifting layer, $Q_1(\boldsymbol{x})$, $Q_2(\boldsymbol{x})$, $\boldsymbol{q}_1(\boldsymbol{x})$ and $\boldsymbol{q}_2(\boldsymbol{x})$ are also often taken as constant parameters, which will be denoted as Q_1, Q_2, $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$, respectively.

To sum up, the integral neural operators can be written as mappings of the form:

$$\mathcal{G}[\boldsymbol{f}; \theta] = \mathcal{Q} \circ \mathcal{L}_L \circ \mathcal{L}_{L-1} \circ \cdots \circ \mathcal{L}_1 \circ \mathcal{P}[\boldsymbol{f}]. \tag{2.6}$$

The architectures of the GKNs, FNOs, NKNs, and our IFNOs mainly differ in the design of their iterative layer update rules in (2.4), which will be elaborated in more detail for each method below. We also summarize their benefits and limitations in Table 1, to highlight that the proposed IFNOs are designed in such a way that all the benefits of these approaches are preserved, while the limitations are overcome.

Graph Kernel Networks (GKNs). As the first integral neural operator, the GKNs introduced in [49] have the foundation in the representation of the solution of a PDE by the Green's function. In the GKNs, it is assumed that the iterative kernel integration part is invariant across layers, i.e.,

$$\mathcal{L}_1 = \mathcal{L}_2 = \cdots = \mathcal{L}_L := \mathcal{L}^{GKN},$$

with the update of each layer network given by

$$\boldsymbol{h}(\boldsymbol{x}, (l+1)\Delta t) = \mathcal{L}^{GKN}[\boldsymbol{h}(\boldsymbol{x}, l\Delta t)] := \sigma\left(W\boldsymbol{h}(\boldsymbol{x}, l\Delta t) + \int_\Omega \kappa(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{y}); \boldsymbol{v})\boldsymbol{h}(\boldsymbol{y}, l\Delta t)d\boldsymbol{y} + \boldsymbol{c}\right). \tag{2.7}$$

Here, σ is an activation function, $W \in \mathbb{R}^{d\times d}$ and $\boldsymbol{c} \in \mathbb{R}^d$ are the learnable tensors, and $\kappa \in \mathbb{R}^{d\times d}$ is a tensor kernel function that takes the form of a (usually shallow) NN whose parameters $\boldsymbol{v}$ are to be learned. The GKN resembles the original ResNet block [82], where the usual discrete affine transformation is substituted by a continuous integral operator. Therefore, the learnt network parameters are resolution-independent: the learned W, $\boldsymbol{c}$, and $\boldsymbol{v}$ are close to optimal even when used with different resolutions, i.e., with different partitions/discretizations of the domain Ω. However, despite its advantage on resolution-independence, in the presence of complex learning tasks the applicability of the GKNs may become compromised by two factors. First, in the most general version of the GKNs, the integral in (2.7) is realized through a message passing graph neural network architecture on a fully-connected graph. Therefore, the GKNs are generally much more expensive than other integral neural operators, say, FNOs, making the GKNs less favorable for large-scale problems. Second, although single-layer and shallow GKNs have been shown to be successful in learning governing equations, e.g., the Darcy [49] and Burgers [50] equations, it was found in [52] that the GKNs may become unstable when the number of its layers increases. As the GKN becomes deeper, either there is no gain in accuracy or increasing values of the loss function occur.

Nonlocal Kernel Networks (NKNs). As a deeper and stabilized modification of the GKNs, the NKNs are introduced in [52] to handle both learning governing equations and classifying images tasks. The NKN stems from the interpretation of the neural network as a discrete nonlocal diffusion reaction equation that, in the limit of infinite layers, is equivalent to a parabolic nonlocal equation. Therefore, its stability in the deep layer limit can be analyzed via nonlocal vector calculus. In the NKNs, the iterative kernel integration is also assumed to be layer-independent. Differs from the GKNs where the next layer representation is defined via a nonlinear operator, the increment of each layer network representation is defined as a nonlinear operator in the NKNs. In particular, the network hidden layer update is given as

$$\begin{aligned}\boldsymbol{h}(\boldsymbol{x},(l+1)\Delta t) &= \mathcal{L}^{NKN}[\boldsymbol{h}(\boldsymbol{x},l\Delta t)]\\ &:= \boldsymbol{h}(\boldsymbol{x},l\Delta t) + \Delta t\left(\int_\Omega \kappa(\boldsymbol{x},\boldsymbol{y},\boldsymbol{f}(\boldsymbol{x}),\boldsymbol{f}(\boldsymbol{y});\boldsymbol{v})(\boldsymbol{h}(\boldsymbol{y},l\Delta t)-\boldsymbol{h}(\boldsymbol{x},l\Delta t))d\boldsymbol{y} - W(\boldsymbol{x};\boldsymbol{w})\boldsymbol{h}(\boldsymbol{x},l\Delta t)+\boldsymbol{c}\right). \quad (2.8)\end{aligned}$$

As for the GKNs, the kernel tensor function $\kappa \in \mathbb{R}^{d\times d}$ is modeled by a NN parameterized by $\boldsymbol{v}$. The reaction term $W \in \mathbb{R}^{d\times d}$ is modeled by another NN parameterized by $\boldsymbol{w}$. The NKN architecture preserves the continuous, integral treatment of the interactions between nodes that characterizes the GKNs, and hence enables resolution independence with respect to the inputs. On the other hand, by modeling the layer representation increment and identifying the number of layers with the number of time steps in a time-discretization scheme, the training of deep NNs in the NKNs is accelerated via the shallow-to-deep technique [83]. In particular, it is obvious to see that by diving both sides of (2.8) by Δt, the term $(\boldsymbol{h}(\cdot,(l+1)\Delta t)-\boldsymbol{h}(\cdot,l\Delta t))/\Delta t$ corresponds to the discretization of a first-order derivative so that this architecture can be interpreted as a nonlinear differential equation in the limit of deep layers, i.e., as $\Delta t \to 0$. Thus, the optimal parameters ($\boldsymbol{v}$, $\boldsymbol{w}$ and $\boldsymbol{c}$) of a shallow network are interpolated and will be reused in a deeper one

as initial guesses. In [52], it is found that the NKNs generalize well to different resolutions and stays stable when the network is getting deeper.

Similarly to the GKNs, since the building blocks of the NKNs are integral operators characterized by space dependent kernels with minimal assumptions, they come at the price of a higher computational cost compared to other networks whose kernels have a convolutional structure (e.g., the standard CNN and FNO). Hence, the NKNs are computationally more expensive than the FNOs, and generally less favorable in large-scale learning tasks.

Fourier Neural Operators (FNOs). The Fourier neural operator (FNO) was first proposed in [51], where the integral kernel κ is parameterized in the Fourier space. In particular, the FNO drops the dependence of kernel κ on the input $\boldsymbol{b}$ and assumes that $\kappa(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{v}) := \kappa(\boldsymbol{x} - \boldsymbol{y}; \boldsymbol{v})$. The integral operator in (2.7) then becomes a convolution operator so that κ can be parameterized directly in the Fourier space. The corresponding $l-$th layer update is then given by

$$\boldsymbol{h}(\boldsymbol{x}, (l+1)\Delta t) = \mathcal{L}_{l+1}^{FNO}[\boldsymbol{h}(\boldsymbol{x}, l\Delta t)] := \sigma\left(W_l \boldsymbol{h}(\boldsymbol{x}, l\Delta t) + \mathcal{F}^{-1}[\mathcal{F}[\kappa(\cdot; \boldsymbol{v}_l)] \cdot \mathcal{F}[\boldsymbol{h}(\cdot, l\Delta t)]](\boldsymbol{x}) + \boldsymbol{c}_l(\boldsymbol{x})\right), \quad (2.9)$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier transform and its inverse, respectively. In practice, $\mathcal{F}$ and $\mathcal{F}^{-1}$ are computed using the the FFT algorithm and its inverse to each component of $\boldsymbol{h}$ separately, with the highest modes truncated and keeping only the first k modes. $\boldsymbol{c}_l(\boldsymbol{x})$ defines a pointwise bias, which is often taken as a constant bias $\boldsymbol{c}_l(\boldsymbol{x}) \equiv \boldsymbol{c}_l$ (see, e.g., [81, 84]). Therefore, $\mathcal{F}[\boldsymbol{h}(\cdot, l\Delta t)]$ has the shape $d \times k$, and the trainable parameters for each hidden layer will be $\boldsymbol{c}_l \in \mathbb{R}^d$, $W_l \in \mathbb{R}^{d\times d}$, and $\mathcal{F}[\kappa(\cdot; \boldsymbol{v}_l)] := R_l \in \mathbb{C}^{d\times d\times k}$. Here, we use W_l, $\boldsymbol{c}_l$ and $\boldsymbol{v}_l$ to highlight the fact that in the FNOs, each layer has different parameters (i.e., different kernels, weights and biases). This is different from the layer-independent kernel in the GKNs and NKNs, and makes the total number of trainable parameters in the FNOs as $DOF^{FNO} := [d(1 + d_F)] + [L(d + d^2 + 2d^2k)] + [d_Q(d + d_u + 1) + d_u]$. Here, the first part is the number of parameters associated with the lifting layer, the second part is associated with the L iterative kernel integration layers, and the last part comes from the projection layer. As the network gets deeper, the second part dominates the total number of parameters, and therefore, the number of trainable parameters in the FNOs grows almost linearly with the increase of L.

Comparing with the GKN and NKN, the FNO has superior efficiency because one can use the FFT to compute (2.9). Moreover, in [84], Kovachiki et al. have proved that with sufficiently large depth L, the FNOs are universal in the sense that they can approximate any continuous operator to a desired accuracy. However, the number of trainable parameters in the FNOs increases as the network gets deeper, which makes the training process of the FNOs more challenging and potentially prone to over-fitting. In [52], it was found that when the network gets deeper, the training error decreases in the FNO while the test error becomes much larger than the training error, indicating that the network is overfitting the training data. Furthermore, if one further increases the number of hidden layer L, training the FNOs becomes challenging

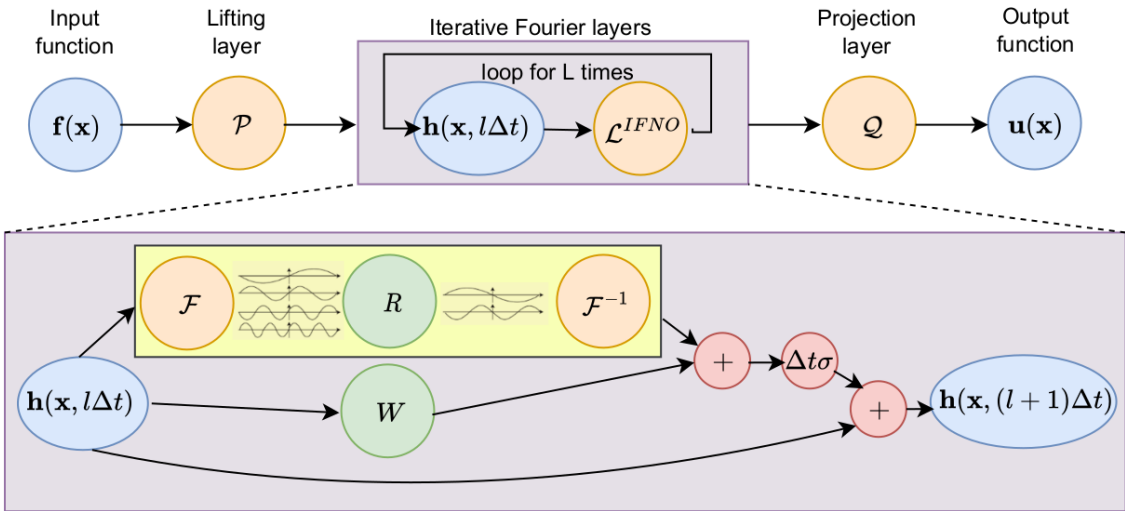

Figure 1: The architecture of IFNO: start from input $\boldsymbol{f}(\boldsymbol{x})$, then 1) Lift to a high dimensional feature space by the lifting layer $\mathcal{P}$ and obtain the first hidden layer representation $\boldsymbol{h}(\boldsymbol{x}, 0)$; 2) Apply L iterative layers with the formulation proposed in (3.4); 3) Project the last hidden layer representation $\boldsymbol{h}(\boldsymbol{x}, L\Delta t)$ back to the target dimension by a shallow network $\mathcal{Q}$.

due to the vanishing gradient phenomenon. On the other hand, as reported in [68], the vanilla version of the FNO is generally restricted to simple geometries and structured data. Although the FNO has superior efficiency and is a theoretically proved universal approximator, its application is generally limited to the cases when the data is structured and less complex such that a shallow network would be sufficient.

3. Implicit Fourier Neural Operators (IFNOs)

To overcome the limitations of the architectures mentioned in Section 2.2, we propose Implicit Fourier Neural Operators (IFNOs), an efficient, deep, and stable integral neural operator for solution operator learning problems. In particular, we first formulate the solution operator as an implicitly defined mapping, and then propose to model it as a fixed point, not via an explicit mapping. Based on this idea, we provide the hidden layer network formulation for the IFNO and illustrate the shallow-to-deep training technique. While the former reduces the number of trainable parameters and memory cost, the latter aims to resolve the difficulty of network training in the limit of deep layers. Finally, we discuss the expressiveness of the IFNOs by showing that as far as there exists a converging fixed point equation for the target implicit problem, the IFNOs would be universal. In the present study, we assume that the datum are structured so the FFT can be employed, and we also note that when the problem domain Ω and the discretization χ are not structured, one might employ the nonlinear mapping extension technique developed in [68] to obtain a structured datum so that the IFNO, based on the following discussion, is still applicable.

3.1. The network architecture

We now propose an IFNO for the solution of the problem outlined in (2.1). To see a guiding principle for our architecture, let us consider the following boundary displacement example:

$$\begin{aligned} \mathcal{K}_{\boldsymbol{b}}[\boldsymbol{u}](\boldsymbol{x}) &= \boldsymbol{g}(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \\ \boldsymbol{u}(\boldsymbol{x}) &= \boldsymbol{u}_D(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial\Omega, \end{aligned} \tag{3.1}$$

where $\mathcal{K}_{\boldsymbol{b}}$ is a differential operator depending on the (possibly nonlinear) material constitutive law, and $\boldsymbol{u}_D$ is the prescribed displacement on the boundary. Given a discretization of the domain defined as $\chi = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_M\}$, the desired network output, or equivalently the numerical solution of (2.1), is then $\boldsymbol{U} = [\boldsymbol{U}_1, \boldsymbol{U}_2, \cdots, \boldsymbol{U}_M] \approx [\boldsymbol{u}(\boldsymbol{x}_1), \cdots, \boldsymbol{u}(\boldsymbol{x}_M)]$. Here, we assume, without loss of generality, that the first β number of points are on $\partial\Omega$, and therefore the solution $\boldsymbol{U}$ on these points is prescribed by the displacement boundary condition $\boldsymbol{u}_D$. With proper discretization methods, such as the finite difference method, for the differential operator $\mathcal{K}_{\boldsymbol{b}}$ and an instance of input vector $\boldsymbol{F} = [\boldsymbol{b}(\boldsymbol{x}_1), \cdots, \boldsymbol{b}(\boldsymbol{x}_M), \boldsymbol{g}(\boldsymbol{x}_1), \cdots, \boldsymbol{g}(\boldsymbol{x}_M), \boldsymbol{u}(\boldsymbol{x}_1), \cdots, \boldsymbol{u}(\boldsymbol{x}_\beta), \boldsymbol{0}, \cdots, \boldsymbol{0}]$, the numerical solution $\boldsymbol{U}$ is determined from the following implicit system of equations:

$$\mathcal{H}(\boldsymbol{U}; \boldsymbol{F}) := \begin{bmatrix} \boldsymbol{U}_1 - \boldsymbol{u}_D(\boldsymbol{x}_1) \\ \vdots \\ \boldsymbol{U}_\beta - \boldsymbol{u}_D(\boldsymbol{x}_\beta) \\ \mathcal{K}^h_{\boldsymbol{b}}(\boldsymbol{U}) - \boldsymbol{G} \end{bmatrix} = \boldsymbol{0}. \tag{3.2}$$

Herein, $\boldsymbol{G} := [\boldsymbol{g}(\boldsymbol{x}_{\beta+1}), \cdots, \boldsymbol{g}(\boldsymbol{x}_M)]$ is the loading term, and $\mathcal{K}^h_{\boldsymbol{b}}$ is the discretized operator. To solve for $\boldsymbol{U}$ from the nonlinear system in (3.2), one can employ fixed-point iteration methods, such as its special case – the Newton-Raphson method. In particular, with an initial guess of the solution (denoted as $\boldsymbol{U}^0$), the process is repeated to produce successively better approximations to the roots of (3.2) following:

$$\boldsymbol{U}^{l+1} = \boldsymbol{U}^l - (\nabla\mathcal{H}(\boldsymbol{U}^l; \boldsymbol{F}))^{-1}\mathcal{H}(\boldsymbol{U}^l; \boldsymbol{F}) := \boldsymbol{U}^l + \mathcal{R}(\boldsymbol{U}^l, \boldsymbol{F}), \tag{3.3}$$

until a sufficiently precise value is reached. Here, we noticed that for each implicit problem, there are infinite numbers of the corresponding fixed point equations, and (3.3) is just one example. In fact, the fixed point method solves the implicit system as long as there exists one fixed point equation with a convergent and unique solution.

Guided by the representation in (3.3), we argue that the desired network output is more aptly described implicitly, and propose to develop a network architecture to model the operator $\mathcal{R}$ and mimic the fixed point method by design. Using the same notations of Section 2, we propose the following iterative network update

formulation

$$\begin{aligned}\boldsymbol{h}(\boldsymbol{x},(l+1)\Delta t) =&\mathcal{L}^{IFNO}[\boldsymbol{h}(\boldsymbol{x},l\Delta t)]\\ :=&\boldsymbol{h}(\boldsymbol{x},l\Delta t)+\Delta t\sigma\left(W\boldsymbol{h}(\boldsymbol{x},l\Delta t)+\mathcal{F}^{-1}[\mathcal{F}[\kappa(\cdot;\boldsymbol{v})]\cdot\mathcal{F}[\boldsymbol{h}(\cdot,l\Delta t)]](\boldsymbol{x})+\boldsymbol{c}(\boldsymbol{x})\right).\end{aligned}\tag{3.4}$$

Note that although the FFT is still applied to each component of $\boldsymbol{h}$ separately with the highest modes truncated as for the FNOs, the hidden layer parameters are taken to be layer-independent, which is distinctly different from the FNOs. Following the conventions in the FNOs, we also take the bias $\boldsymbol{c}_l(\boldsymbol{x})$ as a constant bias ($\boldsymbol{c}(\boldsymbol{x})\equiv\boldsymbol{c}$) in all the subsequent numerical tests. Therefore, the set of trainable parameters in our IFNOs are $P\in\mathbb{R}^{d\times d_F}$ and $\boldsymbol{p}\in\mathbb{R}^d$ for the lifting layer, $Q_1\in\mathbb{R}^{d_Q\times d}$, $Q_2\in\mathbb{R}^{d_u\times d_Q}$, $\boldsymbol{q}_1\in\mathbb{R}^{d_Q}$ and $\boldsymbol{q}_2\in\mathbb{R}^{d_u}$ for the projection layer, and $\boldsymbol{c}\in\mathbb{R}^d$, $W\in\mathbb{R}^{d\times d}$ and $\mathcal{F}(\kappa(\cdot;\boldsymbol{v}))=R\in\mathbb{C}^{d\times d\times k}$ for the hidden layers. The total number of trainable parameters is $DOF^{IFNO}:=[d(1+d_F)]+[d+d^2+2d^2k]+[d_Q(d+d_u+1)+d_u]$, which is independent of the number of hidden layers L, alleviating the major bottleneck of the overfitting issue encountered by the original FNOs with a deeper network. Moreover, this feature also enables the straightforward application of the shallow-to-deep initialization technique.

As the layer becomes deep ($\Delta t\to 0$), (3.4) can be seen as an analog of a discretized ordinary differential equations (ODEs). This allows us to exploit the shallow-to-deep learning technique described in Section 2 for the NKNs. Similarly to in (2.8), we can reinterpret the network update as the time discretization of a differential equation and use the optimal parameters obtained with L layers as the initial guesses for deeper networks. Specifically, let W, $\boldsymbol{c}$ and R be the optimal network parameters obtained by training an IFNO of depth L. For further improving the accuracy of the network, we can increase the number of layers (or equivalently, time steps), and train a new network of depth $\widetilde{L}>L$. The idea of the shallow-to-deep technique is to perform interpolation in time (or across layers) over the optimal parameters obtained at depth L and to scale them in such a way that the final time of the differential equation remains unchanged. In our specific setting, due to the fact that the network parameters are not time dependent, this technique simply corresponds to the initialization of the (deeper) $\widetilde{L}$-layer network by W, $\boldsymbol{c}$ and R.

As a further note, we point out that although the idea of using repeated hidden layers has not been explored for the FNOs, resembling fixed-point methods is not new for neural networks. In [69–72], implicit networks are introduced as an analog to a forward pass through an "infinite depth" network, without storing the intermediate quantities of the forward pass for back-propagation, and hence can be trained using constant memory costs with respect to depth. One can see that our IFNO architecture requires only constant memory cost, similar to implicit networks. Moreover, it preserves the continuous, integral treatment of the interactions between nodes that characterizes integral neural operators. Therefore, the IFNO provides a new and efficient implicit-type neural operator architecture – that is why it is named "implicit".

Table 1 summarizes relevant properties of the IFNOs in comparison with other integral neural operators. In summary, being a resemblance of an implicit equation solver and stable in the limit of deep layers make

the proposed IFNO's architecture a viable tool for modeling problems with complex material responses, since these problems can be considered as PDE solution operator learning tasks.

3.2. Universal approximation properties

In this section, we show that the IFNOs are universal solution finding operators, in the sense that they can approximate a fixed point method to a desired accuracy. Without loss of generality, we consider a 1D domain $\Omega \subset \mathbb{R}$, $\boldsymbol{f}(\boldsymbol{x}) := [\boldsymbol{x}, \hat{\boldsymbol{f}}(\boldsymbol{x})] \in \mathbb{R}^2$ and $\boldsymbol{u}(\boldsymbol{x}) \in \mathbb{R}$. The function $\boldsymbol{u} \in C(\Omega)$ is evaluated at uniformly distributed nodes $\chi = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_M\}$. Let us denote $\boldsymbol{U}^* := [\boldsymbol{u}(\boldsymbol{x}_1), \boldsymbol{u}(\boldsymbol{x}_2), \dots, \boldsymbol{u}(\boldsymbol{x}_M)]$ as the solution we seek, $\boldsymbol{U}^0 := [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_M]$ as the initial guess, $\boldsymbol{C} = [\boldsymbol{c}(\boldsymbol{x}_1), \cdots, \boldsymbol{c}(\boldsymbol{x}_M)]$ as the collection of pointwise bias vectors $\boldsymbol{c}(\boldsymbol{x}_i)$, and $\boldsymbol{F} := [\hat{\boldsymbol{f}}(\boldsymbol{x}_1), \hat{\boldsymbol{f}}(\boldsymbol{x}_2), \dots, \hat{\boldsymbol{f}}(\boldsymbol{x}_M)]$ as the loading vector. We aim to show that for any desired accuracy $\varepsilon > 0$, one can find a sufficiently large $L > 0$ and a set of parameters $\theta_\varepsilon = \{P, \boldsymbol{p}, Q_1, Q_2, \boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{C}, W, R\}$, such that the resultant IFNO model satisfies

$$\left|\left| \mathcal{Q} \circ (\mathcal{L}^{IFNO})^L \circ \mathcal{P}([\boldsymbol{U}^0, \boldsymbol{F}]^{\mathrm{T}}) - \boldsymbol{U}^* \right|\right| \leq \varepsilon, \quad \forall \boldsymbol{F} \in \mathbb{R}^M.$$

Here, the matrix and vector parameters in the lifting and projection layers are taken as pointwise functions. With a slight abuse of notation, we denote $P \in \mathbb{R}^{dM \times d_F M}$ as the collection of the pointwise weight matrices at each discretization point in χ, and a similar convention applies for other matrix and vector parameters in the lifting and projection layers. Hence, the dimension of all trainable parameters are: $\boldsymbol{C} \in \mathbb{R}^{dM}$, $W \in \mathbb{R}^{d \times d}$, $R \in \mathbb{C}^{d \times d \times k}$, $P \in \mathbb{R}^{dM \times d_F M}$, $\boldsymbol{p} \in \mathbb{R}^{dM}$, $Q_1 \in \mathbb{R}^{d_Q M \times dM}$, $Q_2 \in \mathbb{R}^{d_u M} \times d_Q M$, $\boldsymbol{q}_1 \in \mathbb{R}^{d_Q M}$ and $\boldsymbol{q}_2 \in \mathbb{R}^{d_u M}$. With the assumption that $\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^2$ and $\boldsymbol{u}(\boldsymbol{x}) \in \mathbb{R}$, we note that $d_F = 2$ and $d_u = 1$. As will be seen in the proof below, we will further take $d_Q = d$. For the simplicity of notation, in this section we organize the feature vector $\boldsymbol{H} \in \mathbb{R}^{dM}$ in a way such that the components corresponding to each discretization point are adjacent, i.e., $\boldsymbol{H} = [\boldsymbol{H}(\boldsymbol{x}_1), \cdots, \boldsymbol{H}(\boldsymbol{x}_M)]$ and $\boldsymbol{H}(\boldsymbol{x}_i) \in \mathbb{R}^d$. For simplicity, we further assume that the Fourier coefficient is not truncated, and all available frequencies will be used. We point out that under this circumstance, we have $k = M$ and the (discretized) iterative layer can be written as

$$\begin{aligned}
\mathcal{L}^{IFNO}[\boldsymbol{H}(l\Delta t)] =& \boldsymbol{H}(l\Delta t) + \Delta t \sigma \left(\tilde{W} \boldsymbol{H}(l\Delta t) + \mathrm{Re}(\mathcal{F}_{\Delta x}^{-1}(R \cdot \mathcal{F}_{\Delta x}(\boldsymbol{H}(l\Delta t)))) + \boldsymbol{C} \right) \\
=& \boldsymbol{H}(l\Delta t) + \Delta t \sigma \left(V^{IFNO} \boldsymbol{H}(l\Delta t) + \boldsymbol{C} \right),
\end{aligned}$$

with

$$V^{IFNO} := \mathrm{Re} \begin{bmatrix}
\sum_{n=0}^{M-1} R_{n+1} + W & \sum_{n=0}^{M-1} R_{n+1} \exp(\frac{2i\pi\Delta xn}{M}) & \dots & \sum_{n=0}^{M-1} R_{n+1} \exp(\frac{2i\pi(M-1)\Delta xn}{M}) \\
\sum_{n=0}^{M-1} R_{n+1} \exp(\frac{2i\pi\Delta xn}{M}) & \sum_{n=0}^{M-1} R_{n+1} + W & \dots & \sum_{n=0}^{M-1} R_{n+1} \exp(\frac{2i\pi(M-2)\Delta xn}{M}) \\
\vdots & \vdots & \ddots & \vdots \\
\sum_{n=0}^{M-1} R_{n+1} \exp(\frac{2i\pi(M-1)\Delta xn}{M}) & \sum_{n=0}^{M-1} R_{n+1} \exp(\frac{2i\pi(M-2)\Delta xn}{M}) & \dots & \sum_{n=0}^{M-1} R_{n+1} + W
\end{bmatrix}.$$

Here, $R \in \mathbb{C}^{M\times d\times d}$ with $R_i \in \mathbb{C}^{d\times d}$ being the component associated with each discretization point $\boldsymbol{x}_i \in \chi$, $V^{IFNO} \in \mathbb{R}^{dM\times dM}$, $\boldsymbol{C} \in \mathbb{R}^{dM}$, $\tilde{W} := W \oplus W \oplus \cdots \oplus W$ is a $dM \times dM$ block diagonal matrix formed by $W \in \mathbb{R}^{d\times d}$, $\mathcal{F}_{\Delta x}$ and $\mathcal{F}_{\Delta x}^{-1}$ denote the discrete Fourier transform and its inverse, respectively. By further taking $R_2 = \cdots = R_M = W = 0$, a $d \times d$ matrix with all its elements being zero, it suffices to show the universal approximation property for an iterative layer as follows:

$$\mathcal{L}^{IFNO}(\boldsymbol{H}(l\Delta t)) := \boldsymbol{H}(l\Delta t) + \Delta t\sigma\left(\tilde{V}\boldsymbol{H}(l\Delta t) + \boldsymbol{C}\right)$$

where $\tilde{V} := \mathbf{1}_{[M,M]} \otimes V$ with $V \in \mathbb{R}^{d\times d}$ and $\mathbf{1}_{[m,n]}$ being an m by n all-ones matrix.

Before stating our main theoretical results, we need the following assumptions on $\boldsymbol{U}^*$ and $\mathcal{R}$:

Assumption 1. There exists a fixed point equation, $\boldsymbol{U} = \boldsymbol{U} + \mathcal{R}(\boldsymbol{U}, \boldsymbol{F})$ for the implicit problem (3.2), such that $\mathcal{R} : \mathbb{R}^{2M} \mapsto \mathbb{R}^M$ is a continuous function satisfying $\mathcal{R}(\boldsymbol{U}^*, \boldsymbol{F}) = \mathbf{0}$ and $||\mathcal{R}(\hat{\boldsymbol{U}}, \boldsymbol{F}) - \mathcal{R}(\tilde{\boldsymbol{U}}, \boldsymbol{F})||_{l^2(\mathbb{R}^M)} \leq m||\hat{\boldsymbol{U}} - \tilde{\boldsymbol{U}}||_{l^2(\mathbb{R}^M)}$ for any two vectors $\hat{\boldsymbol{U}}, \tilde{\boldsymbol{U}} \in \mathbb{R}^M$. Here, $m > 0$ is a constant independent of $\boldsymbol{F}$.

Assumption 2. With the initial guess $\boldsymbol{U}^0 := [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_M]$, the fixed-point iteration

$$\boldsymbol{U}^{l+1} = \boldsymbol{U}^l + \mathcal{R}(\boldsymbol{U}^l, \boldsymbol{F}), \quad l = 0, 1, \ldots.$$

converges, i.e., for any given $\varepsilon > 0$, there exists an integer L such that

$$||\boldsymbol{U}^l - \boldsymbol{U}^*||_{l^2(\mathbb{R}^M)} \leq \varepsilon, \quad \forall l > L,$$

for all possible input instances $\boldsymbol{F} \in \mathbb{R}^M$ and their corresponding solutions $\boldsymbol{U}^*$.

Next, we prove that the IFNOs are universal, i.e., give a fixed point method and solution $\boldsymbol{U}^*$ satisfying Assumptions 1-2, one can find an IFNO whose output approximates $\boldsymbol{U}^*$ to a desired accuracy, $\varepsilon > 0$. To be more precise, we will prove the following theorem:

Theorem 1 (Universal approximation)**.** Let $\boldsymbol{U}^* = [\boldsymbol{u}(\boldsymbol{x}_1), \boldsymbol{u}(\boldsymbol{x}_2), \ldots, \boldsymbol{u}(\boldsymbol{x}_M)]$ be the ground-truth solution that satisfies Assumptions 1-2, the activation function σ for all iterative kernel integration layers be the ReLU function, and the activation function in the projection layer be the identity function. Then for any $\varepsilon > 0$, there exist sufficiently large layer number $L > 0$ and feature dimension number $d > 0$, such that one can find a parameter set $\theta_\varepsilon = \{P, \boldsymbol{p}, Q_1, Q_2, \boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{C}, V\}$ with $P \in \mathbb{R}^{dM\times 2M}, \boldsymbol{p} \in \mathbb{R}^{dM}, Q_1 \in \mathbb{R}^{dM\times dM}, Q_2 \in \mathbb{R}^{M\times dM}, \boldsymbol{q}_1 \in \mathbb{R}^{dM}, \boldsymbol{q}_2 \in \mathbb{R}^M, \boldsymbol{C} \in \mathbb{R}^{dM}, V \in \mathbb{R}^{d\times d}$ with the corresponding IFNO model satisfies

$$\left|\left|\mathcal{Q} \circ (\mathcal{L}^{IFNO})^L \circ \mathcal{P}([\boldsymbol{U}^0, \boldsymbol{F}]^{\mathrm{T}}) - \boldsymbol{U}^*\right|\right| \leq \varepsilon, \quad \forall \boldsymbol{F} \in \mathbb{R}^M.$$

Before proceeding to the proof of this main theorem, we first show the approximation property of a shallow neural network:

Lemma 1. Given a continuous function $\mathcal{T} : \mathbb{R}^{2M} \mapsto \mathbb{R}^M$, and a non-polynomial and continuous activation function σ, for any constant $\hat{\varepsilon} > 0$ there exists a shallow neural network model $\hat{\mathcal{T}} := S\sigma\left(B\boldsymbol{X} + A\right)$ such that

$$||\mathcal{T}(\boldsymbol{X}) - \hat{\mathcal{T}}(\boldsymbol{X})||_{l^2(\mathbb{R}^M)} \leq \hat{\varepsilon}, \quad \forall \boldsymbol{X} \in \mathbb{R}^{2M},$$

for sufficiently large feature dimension $\hat{d} > 0$. Here, $S \in \mathbb{R}^{M \times \hat{d}M}$, $B \in \mathbb{R}^{\hat{d}M \times 2M}$, and $A \in \mathbb{R}^{\hat{d}M}$ are matrices/vectors which are independent of $\boldsymbol{X}$.

Proof. As shown in [85], when σ is non-polynomial and continuous, $\text{span}(\sigma(\boldsymbol{r} \cdot \boldsymbol{X} + a))$ is dense in $C(\mathbb{R}^{2M})$, where $\boldsymbol{r} \in \mathbb{R}^{1 \times 2M}$, and $a \in \mathbb{R}$. Therefore, denoting $\mathcal{T}(\boldsymbol{X}) = [\mathcal{T}_1(\boldsymbol{X}), \ldots, \mathcal{T}_M(\boldsymbol{X})]$, for each $\mathcal{T}_i(\boldsymbol{X}) \in \mathbb{R}$ there exist $\boldsymbol{s}_i \in \mathbb{R}^{1 \times \hat{d}}, B_i \in \mathbb{R}^{\hat{d} \times 2M}$, and $\boldsymbol{a}_i \in \mathbb{R}^{\hat{d}}$, such that

$$|\mathcal{T}_i(\boldsymbol{X}) - \boldsymbol{s}_i\sigma(B_i\boldsymbol{X} + \boldsymbol{a}_i)| \leq \frac{\hat{\varepsilon}}{\sqrt{M}}, \quad \forall \boldsymbol{X} \in \mathbb{R}^{2M}.$$

Let

$$S := \begin{bmatrix} \boldsymbol{s}_1 & \boldsymbol{0} & \ldots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{s}_2 & \ldots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \ldots & \boldsymbol{s}_M \end{bmatrix} \in \mathbb{R}^{M \times \hat{d}M}, \quad B := \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_M \end{bmatrix} \in \mathbb{R}^{\hat{d}M \times 2M}, \quad A = \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \vdots \\ \boldsymbol{a}_M \end{bmatrix} \in \mathbb{R}^{\hat{d}M},$$

we then obtain

$$||\mathcal{T}(\boldsymbol{X}) - S\sigma\left(B\boldsymbol{X} + A\right)||_{l^2(\mathbb{R}^M)} = \sqrt{\sum_{i=1}^{M} |\mathcal{T}_i(\boldsymbol{X}) - \boldsymbol{s}_i\sigma(B_i\boldsymbol{X} + \boldsymbol{a}_i)|^2} \leq \sqrt{M \times \frac{\hat{\varepsilon}^2}{M}} = \hat{\varepsilon},$$

for all $\boldsymbol{X} \in \mathbb{R}^{2M}$. □

We now proceed to the proof of Theorem 1:

Proof. Since $\boldsymbol{U}^*$ satisfies Assumptions 1-2, for any $\varepsilon > 0$, we first pick a sufficiently large integer L such that $||\boldsymbol{U}^L - \boldsymbol{U}^*||_{l^2(\mathbb{R}^M)} \leq \frac{\varepsilon}{2}$. By taking $\hat{\varepsilon} := \frac{m\varepsilon}{2(1+m)^L}$ in Lemma 1, there exists a sufficiently large feature dimension $\hat{d}$ and one can find $S \in \mathbb{R}^{M \times \hat{d}M}$, $B \in \mathbb{R}^{\hat{d}M \times 2M}$, and $A \in \mathbb{R}^{\hat{d}M}$, such that $\hat{\mathcal{R}}(\boldsymbol{U}, \boldsymbol{F}) := S\sigma(B[\boldsymbol{U}, \boldsymbol{F}]^{\mathrm{T}} + A)$ satisfies

$$||\mathcal{R}(\boldsymbol{U}, \boldsymbol{F}) - \hat{\mathcal{R}}(\boldsymbol{U}, \boldsymbol{F})||_{l^2(\mathbb{R}^M)} = ||\mathcal{R}(\boldsymbol{U}, \boldsymbol{F}) - S\sigma(B[\boldsymbol{U}, \boldsymbol{F}]^{\mathrm{T}} + A)||_{l^2(\mathbb{R}^M)} \leq \hat{\varepsilon} = \frac{m\varepsilon}{2(1+m)^L},$$

where m is the contraction parameter of $\mathcal{R}$, as defined in Assumption 1. By this construction, we know that S has independent rows. Denoting $\tilde{d} := \hat{d} + 1 > 0$, there exists the right inverse of S, which we denote as $S^+ \in \mathbb{R}^{(\tilde{d}-1)M \times M}$, such that

$$SS^+ = I_M, \quad S^+S := \tilde{I}_{(\tilde{d}-1)M},$$

where I_M is the M by M identity matrix, $\tilde{I}_{(\tilde{d}-1)M}$ is a $(\tilde{d}-1)M$ by $(\tilde{d}-1)M$ block matrix with each of its element being either 1 or 0. Hence, for any vector $Z \in \mathbb{R}(\tilde{d}-1)M$, we have $\sigma(\tilde{I}_{(\tilde{d}-1)M}Z) = \tilde{I}_{(\tilde{d}-1)M}\sigma(Z)$. Moreover, we note that S has a very special structure: from the $((i-1)(\tilde{d}-1)+1)$-th to the $(i(\tilde{d}-1))$-th column of S, all nonzero elements are on its i-th row. Correspondingly, we can also choose S^+ to have a special structure: from the $((i-1)(\tilde{d}-1)+1)$-th to the $(i(\tilde{d}-1))$-th row of S^+, all nonzero elements are on its i-th column. Hence, when multiplying S^+ with $\boldsymbol{U}$, there will be no entanglement between different components of $\boldsymbol{U}$. That means, S^+ can be seen as a pointwise weight function.

We now construct the IFNO as follows. In this construction, we choose the feature dimention as $d := \tilde{d}M$. With the input $[\boldsymbol{U}^0, \boldsymbol{F}] \in \mathbb{R}^{2M}$, for the lift layer we set

$$P := \mathbf{1}_{[M,1]} \otimes \begin{bmatrix} S^+ & \mathbf{0} \\ \mathbf{0} & I_M \end{bmatrix} = \underbrace{\begin{bmatrix} S^+ & \mathbf{0} & S^+ & \mathbf{0} & \cdots & S^+ & \mathbf{0} \\ \mathbf{0} & I_M & \mathbf{0} & I_M & \cdots & \mathbf{0} & I_M \end{bmatrix}^{\mathrm{T}}}_{\text{repeated for } M \text{ times}} \in \mathbb{R}^{dM \times 2M}, \quad \text{and} \quad \boldsymbol{p} := \mathbf{0} \in \mathbb{R}^{dM}.$$

As such, the initial layer of feature is then given by

$$\boldsymbol{H}^0 = \mathcal{P}([\boldsymbol{U}^0, \boldsymbol{F}]^{\mathrm{T}}) = \mathbf{1}_{[M,1]} \otimes [S^+\boldsymbol{U}^0, \boldsymbol{F}]^{\mathrm{T}} \in \mathbb{R}^{dM}.$$

Here, we point out that P and $\boldsymbol{p}$ can be seen as pointwise weight and bias functions, respectively.

Next we construct the iterative layer $\mathcal{L}^{INFO}$, by setting

$$V := \begin{bmatrix} \tilde{I}_{(\tilde{d}-1)M}B/M \\ 0 \end{bmatrix} \begin{bmatrix} S/\Delta t & \mathbf{0} \\ \mathbf{0} & I_M/\Delta t \end{bmatrix}, \ \tilde{V} := \mathbf{1}_{[M,M]} \otimes V, \quad \text{and } \boldsymbol{C} := \mathbf{1}_{[M,1]} \otimes \begin{bmatrix} \tilde{I}_{(\tilde{d}-1)M}A/\Delta t \\ \mathbf{0} \end{bmatrix}.$$

Note that $\tilde{V}$ falls into the formulation of V^{IFNO}, by letting $R_1 = V$ and $R_2 = R_2 = \cdots = R_M = W = 0$. For the $l+1$-th layer of feature vector, we then arrive at

$$\begin{aligned}
&\boldsymbol{H}((l+1)\Delta t) = \boldsymbol{H}(l\Delta t) + \Delta t\sigma\left(\tilde{V}\boldsymbol{H}(l\Delta t) + \boldsymbol{C}\right) \\
=&\boldsymbol{H}(l\Delta t) + \left(I_M \otimes \begin{bmatrix} S^+S & \mathbf{0} \\ \mathbf{0} & I_M \end{bmatrix}\right) \sigma\left(\left(\mathbf{1}_{[M,1]} \otimes \begin{bmatrix} B/M \\ \mathbf{0} \end{bmatrix}\right)\left(\mathbf{1}_{[1,M]} \otimes \begin{bmatrix} S & \mathbf{0} \\ \mathbf{0} & I_M \end{bmatrix}\right)\boldsymbol{H}(l\Delta t) + \mathbf{1}_{[M,1]} \otimes \begin{bmatrix} A \\ \mathbf{0} \end{bmatrix}\right),
\end{aligned}$$

where $\boldsymbol{H}(l\Delta t) = [\hat{\boldsymbol{h}}_1^{l\Delta t}, \hat{\boldsymbol{h}}_2^{l\Delta t}, \ldots, \hat{\boldsymbol{h}}_{2M-1}^{l\Delta t}, \hat{\boldsymbol{h}}_{2M}^{l\Delta t}]^{\mathrm{T}}$ denotes the (spatially discretized) hidden layer feature at the $l-$th iterative layer of the IFNO. Subsequently, we note that the second part of the feature vector, $\hat{\boldsymbol{h}}_{2j}^{l\Delta t} \in \mathbb{R}^M$, satisfies

$$\hat{\boldsymbol{h}}_{2j}^{(l+1)\Delta t} = \hat{\boldsymbol{h}}_{2j}^{l\Delta t} = \cdots = \hat{\boldsymbol{h}}_{2j}^{0} = \boldsymbol{F}, \quad \forall l = 0, \cdots, L-1, \forall j = 1, \cdots, M$$

Problem	Data collected from	Input function	Output function
Porous medium I	Darcy's equation	Permeability field	Pressure field
Porous medium II	Darcy's equation	Source field & boundary condition	Pressure field
Fiber-reinforced material	Holzapfel-Gasser-Odgen (HGO) model	Boundary condition	Displacement field
Glass-ceramics fracture	Quasi-static linear peridynamic solid model	Boundary displacement & previous damage field	Damage field
Latex glove sample (palm region)	Digital Image Correlation (DIC) displacement tracking	Boundary condition & previous displacement field	Displacement field

Table 2: Setup for the three numerical examples in Section 4 and the application in Section 5.

Hence, the first part of the feature vector, $\hat{\boldsymbol{h}}_{2j-1}^{l\Delta t} \in \mathbb{R}^{(\tilde{d}-1)M}$, satisfies the following iterative rule:

$$\hat{\boldsymbol{h}}_{2j-1}^{(l+1)\Delta t} = \hat{\boldsymbol{h}}_{2j-1}^{l\Delta t} + S^{+}S\sigma(B[S\hat{\boldsymbol{h}}_{2j-1}^{l\Delta t}, \boldsymbol{F}]^{\mathrm{T}} + A), \quad \forall l = 0, \cdots, L-1, \forall j = 1, \cdots, M,$$

and

$$\hat{\boldsymbol{h}}_{1}^{(l+1)\Delta t} = \hat{\boldsymbol{h}}_{3}^{(l+1)\Delta t} = \cdots = \hat{\boldsymbol{h}}_{2M-1}^{(l+1)\Delta t}.$$

Finally, for the projection layer $\mathcal{Q}$, we set the activation function in the projection layer as the identity function, $Q_1 := I_{dM}$ (the identity matrix of size dM), $Q_2 := [S, \boldsymbol{0}] \in \mathbb{R}^{M \times dM}$, $\boldsymbol{q}_1 := \boldsymbol{0} \in \mathbb{R}^{dM}$, and $\boldsymbol{q}_2 := \boldsymbol{0} \in \mathbb{R}^{M}$. Denoting the output of IFNO as $\boldsymbol{U}_{IFNO} := \mathcal{Q} \circ (\mathcal{L}^{IFNO})^{L} \circ \mathcal{P}([\boldsymbol{U}^0, \boldsymbol{F}]^{\mathrm{T}})$, we now show that $\boldsymbol{U}_{IFNO}$ can approximate $\boldsymbol{U}^*$ with a desired accuracy ε:

$$\begin{aligned}
||\boldsymbol{U}_{IFNO} - \boldsymbol{U}^*|| &\leq ||\boldsymbol{U}_{IFNO} - \boldsymbol{U}^L||_{l^2(\mathbb{R}^M)} + ||\boldsymbol{U}^L - \boldsymbol{U}^*||_{l^2(\mathbb{R}^M)} \\
&\leq ||S\hat{\boldsymbol{h}}_1^{L\Delta t} - \boldsymbol{U}^L||_{l^2(\mathbb{R}^M)} + \frac{\varepsilon}{2} \quad \textit{(by Assumption 2)} \\
&\leq ||S\hat{\boldsymbol{h}}_1^{(L-1)\Delta t} - \boldsymbol{U}^{L-1}||_{l^2(\mathbb{R}^M)} + ||\hat{\mathcal{R}}(S\hat{\boldsymbol{h}}_1^{(L-1)\Delta t}, \boldsymbol{F}) - \mathcal{R}(\boldsymbol{U}^{L-1}, \boldsymbol{F})||_{l^2(\mathbb{R}^M)} + \frac{\varepsilon}{2} \\
&\leq ||S\hat{\boldsymbol{h}}_1^{(L-1)\Delta t} - \boldsymbol{U}^{L-1}||_{l^2(\mathbb{R}^M)} + ||\hat{\mathcal{R}}(S\hat{\boldsymbol{h}}_1^{(L-1)\Delta t}, \boldsymbol{F}) - \mathcal{R}(S\hat{\boldsymbol{h}}_1^{(L-1)\Delta t}, \boldsymbol{F})||_{l^2(\mathbb{R}^M)} \\
&+ ||\mathcal{R}(S\hat{\boldsymbol{h}}_1^{(L-1)\Delta t}, \boldsymbol{F}) - \mathcal{R}(\boldsymbol{U}^{L-1}, \boldsymbol{F})||_{l^2(\mathbb{R}^M)} + \frac{\varepsilon}{2} \\
&\leq (1+m)||S\hat{\boldsymbol{h}}_1^{(L-1)\Delta t} - \boldsymbol{U}^{L-1}||_{l^2(\mathbb{R}^M)} + \frac{m\varepsilon}{2(1+m)^L} + \frac{\varepsilon}{2} \quad \textit{(by Lemma 1 and Assumption 1)} \\
&\leq \frac{m\varepsilon}{2(1+m)^L}(1 + (1+m) + (1+m)^2 + \cdots + (1+m)^{L-1}) + \frac{\varepsilon}{2} \\
&\leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon.
\end{aligned}$$

□

4. Numerical Examples

In this section, we illustrate the performance of the proposed IFNOs on three benchmark material modeling problems: (i) the flow through a porous medium, (ii) the deformation of a hyperelastic and anisotropic fiber-reinforced material, and (iii) the brittle fracture mechanics in glass-ceramics. The detailed settings of

each example, including the choices of high-fidelity (ground-truth) training/testing data generation, input function, and output function, are provided in Table 2. For all numerical experiments, we compare the IFNO to the FNO as the baseline approach, since the other two integral neural operators (i.e., the GKNs and the NKNs) are computationally much more expensive and hence not feasible given the data size and our computational resources. All our numerical experiments were performed on a machine with 2.8 GHz 8-core CPU and a single Nvidia RTX 3060 GPU. For the implementation of the IFNOs and the FNOs, we used the Pytorch package provided in [51]. The optimization was performed with the Adam optimizer. To conduct a fair comparison, for each method, we tuned the hyperparameters, including the learning rates, the decay rates and the regularization parameters, to minimize the validation errors (or the test errors when there is no available validation set). Furthermore, for each example and each method, we repeated the numerical experiment for five different random initializations, and reported the averaged relative mean squared errors and their standard errors. For a compact presentation of the results, we reported the relative mean squared errors in plots, as functions of the number of hidden layers (L), with error bars representing the standard errors over five simulations. A more detailed error comparison is provided in the appendix.

4.1. The flow through a porous medium

We consider the modeling problem of two-dimensional sub-surface flows through a porous medium with heterogeneous permeability field. Following the settings in [49], the high-fidelity synthetic simulation data for this example are described by the Darcy's flow. Here, the physical domain is $D = [0,1]^2$, $b(\boldsymbol{x})$ is the permeability field, and the operator $\mathcal{K}_b$ is then an elliptic operator associated with $b(\boldsymbol{x})$. In particular, the Darcy's equation has the form:

$$\begin{aligned} -\nabla \cdot (b(\boldsymbol{x})\nabla u(\boldsymbol{x})) &= g(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) &= u_D(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial\Omega. \end{aligned}$$

In this context, our goal is to learn the solution operator of the Darcy's equation and compute the pressure field $u(\boldsymbol{x})$. In this example two study scenarios are considered, corresponding to two different real-world application scenarios:

1. (Porous medium I, see Figure 4) Considering a fixed source field $g(\boldsymbol{x}) = 1$ and Dirichlet boundary condition $u_D(\boldsymbol{x}) = 0$, we aim to obtain the pressure field $u(\boldsymbol{x})$ for each permeability field $b(\boldsymbol{x})$. Therefore, the neural operators are employed to learn the mapping from $\boldsymbol{f}(\boldsymbol{x}) := [\boldsymbol{x}, b(\boldsymbol{x})]$ to $u(\boldsymbol{x})$. This setting corresponds to a scenario that the same lab test protocols are applied to heterogeneous material samples with different microstructures, and our learning goal is to predict the material response for a new and unseen sample. Note that this setting is also the benchmark problem considered in a series of integral neural operator studies [49–52].

2. (Porous medium II, see Figure 5) Considering a fixed permeability field $b(\boldsymbol{x})$, we aim to estimate the

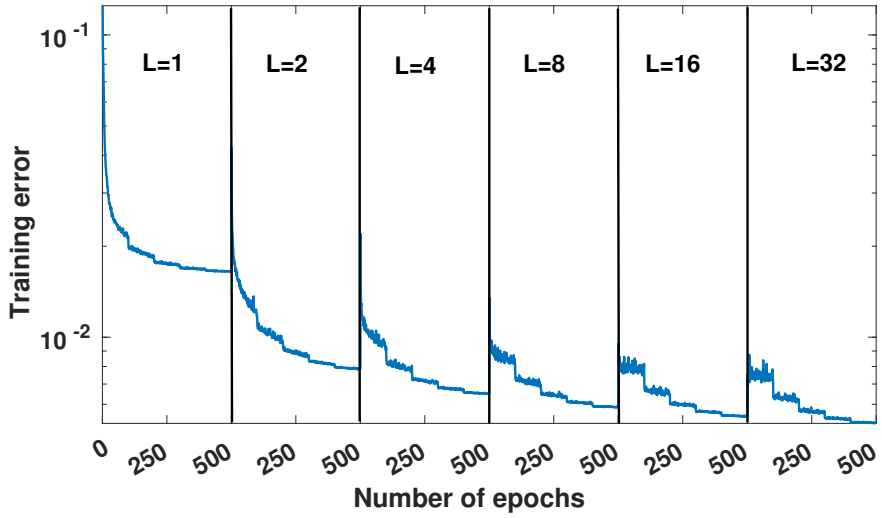

Figure 2: The training error history using the shallow-to-deep technique on the porous medium problem I.

pressure field $u(\boldsymbol{x})$ subject to different source fields $g(\boldsymbol{x})$ and Dirichlet boundary conditions $u_D(\boldsymbol{x})$. That means, the neural operators are employed to learn the mapping from $\boldsymbol{f}(\boldsymbol{x}) := [\boldsymbol{x}, g(\boldsymbol{x}), \tilde{u}_D(\boldsymbol{x})]$ to $u(\boldsymbol{x})$. This setting corresponds to a scenario that different lab tests are available for a given material sample with unknown microstructure, and our learning goal is to predict the mechanical response of this sample under a new and unseen loading. We note that this setting reflects the typical material mechanical testing experiments, see, e.g., [28], where one representative material sample is tested under several loading protocols and the responses, such as the displacement fields and/or stretch-stress curves, are provided.

Model	$L = 1$	$L = 2$	$L = 4$	$L = 8$	$L = 16$	$L = 32$
Number of trainable parameters						
FNO, setting I	171.42k	338.37k	672.26k	1.34M	2.68M	5.35M
IFNO, setting I	171.42k	171.42k	171.42k	171.42k	171.42k	171.42k
FNO, setting II	300.48k	596.45k	1.19M	2.37M	4.74M	9.48M
IFNO, setting II	300.48k	300.48k	300.48k	300.48k	300.48k	300.48k
Training time for each epoch (in second)						
FNO, setting I	0.406±0.017	0.619±0.005	0.910±0.047	1.788±0.014	3.100±0.161	5.694±0.294
IFNO, setting I	0.342±0.002	0.471±0.003	0.730±0.004	1.239±0.008	2.246±0.017	4.300±0.028
FNO, setting II	0.228±0.013	0.498±0.007	0.591±0.014	1.105±0.021	1.725±0.156	3.264±0.187
IFNO, setting II	0.185±0.002	0.256±0.006	0.396±0.012	0.713±0.022	1.281±0.045	2.455±0.049

Table 3: Example 1: the flow of a fluid through a porous medium. The number of trainable parameters and the training time of each epoch with different hidden layer number for each model.

Setting and results of porous medium I. As standard simulations of subsurface flow, the permeability $b(\boldsymbol{x})$ is modeled as a two-valued piecewise constant function with random geometry such that the two values have a ratio of 4. Specifically, we generated $1,100$ samples of $b(\boldsymbol{x})$ according to $b \sim \psi_{\#}\mathcal{N}(0, (-\Delta + 9I)^{-2})$, where ψ

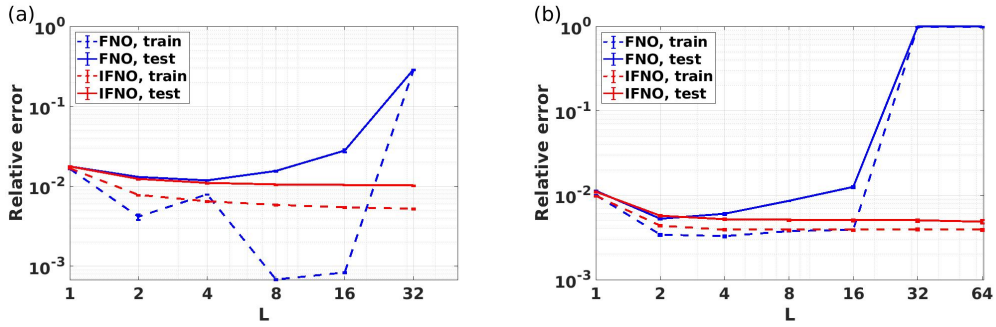

Figure 3: The flow of a fluid through a porous medium (example 1). Comparison of relative mean squared errors of pressure field from FNOs and IFNOs. (a) Results from setting I, where neural operators are employed to solve for the corresponding pressure field for each giving permeability field. (b) Results from setting II, where neural operators are employed to solve for the corresponding pressure field with each given pair of source field $g(\boldsymbol{x})$ and Dirichlet boundary condition $u_D(\boldsymbol{x})$.

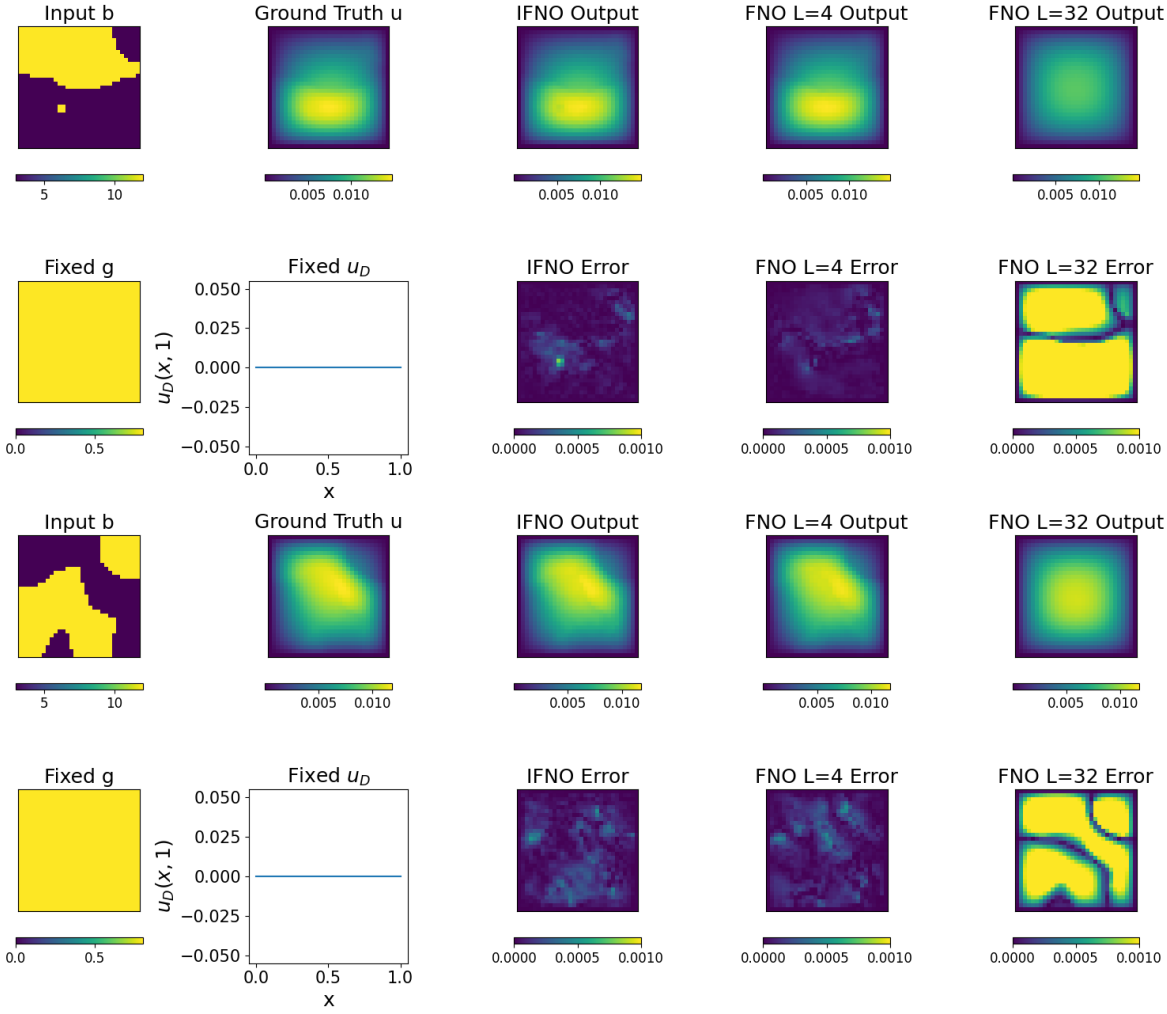

Figure 4: The flow of a fluid through a porous medium setting I, prediction of pressure field from different permeability field $b(\boldsymbol{x})$ and fixed source field $g(\boldsymbol{x}) = 1$ and boundary condition $u_D(\boldsymbol{x}) = 0$ (example 1). A visualization of FNO and IFNO performances on two instances of permeability parameter $b(\boldsymbol{x})$. Here, the best IFNO results ($L = 32$), best FNO results ($L = 4$), and the deepest FNO results ($L = 32$) are reported.

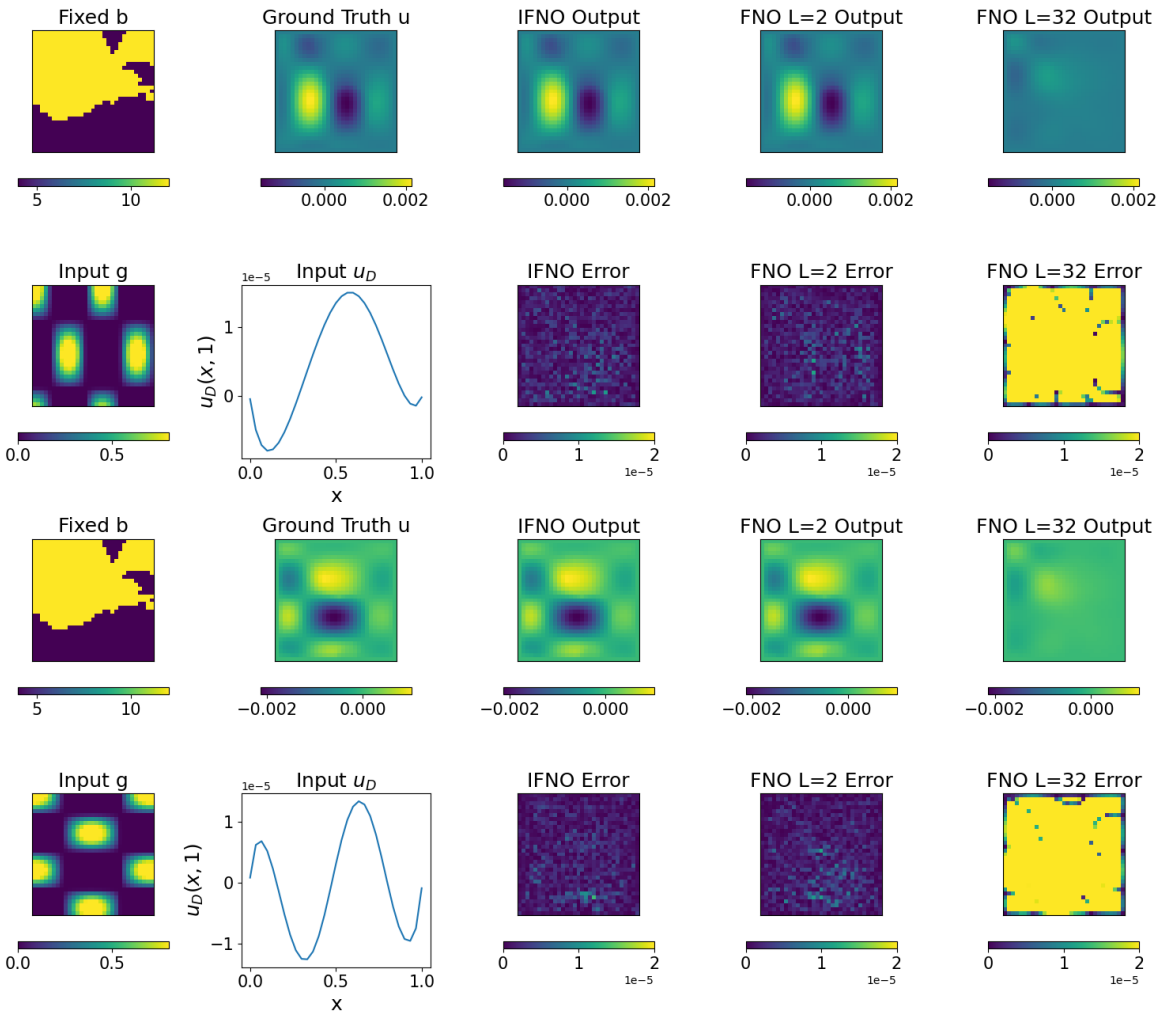

Figure 5: The flow of a fluid through a porous medium setting II, prediction of pressure field on a fixed permeability field $b(\boldsymbol{x})$ and different source field $g(\boldsymbol{x})$ and boundary condition $u_D(\boldsymbol{x})$ (example 1). A visualization of FNO and IFNO performances on two instances of $g(\boldsymbol{x})$ and $u_D(\boldsymbol{x})$. Here, the best IFNO results ($L = 32$), best FNO results ($L = 2$), and the deepest FNO results ($L = 32$) are reported.

takes a value of 12 on the positive part of the real line and a value of 3 on the negative. For cross-validation, the total dataset were divided into a training set with $1,000$ samples and a test dataset with 100 samples. Then, for each sample the high-fidelity solution of u was generated by using a second-order finite difference scheme to solve the Darcy's equation on a 241×241 grid solution, and both the input and output functions were down-sampled to a structured grid χ with grid size $\Delta x = 1/30$. In this experiment, for both the FNOs and IFNOs, we set the dimension of $\boldsymbol{h}$ as $d = 32$, and the number of truncated Fourier modes as $k = 9 \times 9$. For each depth L, we trained the network for 500 epochs with a learning rate of $1e{-}3$, then decrease the learning rate with a ratio 0.5 every 100 epochs. For the IFNOs, the network was trained with the shallow-to-deep training procedure: we initialized the $L-$layer network parameters from the $(L/2)-$layer IFNOs model. To demonstrate the effect of this training technique, a plot of training error is provided in Figure 2 as a function of the number of epochs from $L = 1$ till $L = 32$. One can see that the training error

is consistently decreasing as the network gets deeper. This strategy was also employed for other examples in this paper.

In Figure 3(a) we report the averaged relative mean squared errors from setting I as a function of iterative layer number L; the number of trainable parameters and training time per epoch for each model is provided in Table 3. We can observe that as we increases L to 8 and 16, the FNO reaches a relatively low level of error on the training dataset ($O(10^{-4})$). However, the test error of the FNOs deteriorates with the increase of L, and reaches $O(10^{-2})$ when $L = 8$ or 16. This indicates that the network is overfitting the training data. Moreover, for $L \geq 32$, the training of the FNOs becomes challenging due to the vanishing gradient phenomenon [86]. In contrast, the IFNOs trained with the shallow-to-deep initialization are robust and not subject to the overfitting issues: the test error improves as one increases L, and stays at a similar magnitude as the training error. Comparing with the FNOs with the same number of layers, the IFNOs have a much smaller number of trainable parameters and lower test errors in all $L > 1$ cases. Specifically, the IFNO reaches its best performance when $L = 32$, where the averaged (relative) test error is 1.02%. On the other hand, the lowest error for the FNO is 1.19%, achieved when $L = 4$. In Figure 4, we show the plots of solutions obtained with the best IFNO, the best FNO, and the deepest FNO, in correspondence of two instances of permeability parameter $b(\boldsymbol{x})$. Both the solutions and the errors are plotted, showing that the FNO loses accuracy when the layer gets deeper ($L = 32$), while all other solutions are visually consistent with the ground-truth solutions. Moreover, from Table 3 one can see that the IFNO generally requires a shorter time for the training of each epoch. However, we would like to point out that the overall training time of IFNOs is still generally longer than the standard FNO, due to the additional shallow-to-deep procedure.

Setting and results of porous medium II. In setting II, we considered a fixed realization of permeability field $b(\boldsymbol{x})$, which was generated following the same procedure as described in setting I. In this context, our goal is to predict the pressure field driven by different source fields $g(\boldsymbol{x})$ and Dirichlet boundary conditions u_D. To generate each sample, we set the source field as $g(\boldsymbol{x}) = \cos(2\pi a_x x)\cos(2\pi a_y y)$. Here, a_x and a_y are the constant coefficients randomly generated as $a_x, a_y \sim \mathcal{U}(0.5, 2)$, the uniform distribution on $[0.5, 2]$. To generate the boundary condition u_D, we set the pressure on the top edge of the domain as $u_D(x, 1) = U_0(t_1 \sin(2\pi x) + t_2 \sin(4\pi x))/(t_1 + t_2)$, where $U_0 \sim \mathcal{U}(-0.001, 0.001)$, and $t_1, t_2 \sim \mathcal{U}(0, 1)$. On the rest of boundaries, the pressure was prescribed as $u_D(x, y) = U_0$. For training and cross-validation, we generated 600 samples in total and split it as a training set with 500 samples and a test set with 100 samples. Similar to setting I, the training and test measurements of the pressure fields u were also generated by solving the Darcy's equation and down-sampling to a $M = 31 \times 31$ grid. In this experiment, for both the FNOs and IFNOs, we set the dimension of $\boldsymbol{h}$ as $d = 32$, and the number of truncated Fourier modes as $k = 12 \times 12$. For each depth L, we trained the network for 500 epochs with a learning rate of $3e{-}3$, then decrease the learning rate with a ratio 0.5 every 100 epochs.

In Figures 3(b), we report the relative mean squared errors from each model, with hidden layer number L from 1 to 64. The number of trainable parameters for each model is provided in Table 3. Similarly to

Parameter	c_{10}	K	k_1	k_2	α	κ
Value	0.3846	0.8333	0.1	1.5	$\pi/2$	0

Table 4: Parameter values of the HGO model for data generation in example 2.

Set ID	Protocol	max U_x on the right edge	max U_y on the top edge
1	Biaxial Stretch 1 : 1	0.4	0.4
2	Biaxial Stretch 0.66 : 1	0.4	0.6
3	Biaxial Stretch 0.5 : 1	0.2	0.4
4	Biaxial Stretch 0.33 : 1	0.2	0.6
5	Biaxial Stretch 1 : 0.66	0.6	0.4
6	Biaxial Stretch 1 : 0.5	0.4	0.2
7	Biaxial Stretch 1 : 0.33	0.6	0.2
8	Uniaxial Stretch in x	0.4	0
9	Uniaxial Stretch in y	0	0.4

Table 5: Nine protocols of the synthetic biaxial mechanical testing on a 1×1 hyperelastic and anisotropic fiber-reinforced material sample. All simulations are generated with FEniCS [87] based on the HGO model.

the porous medium setting I, when increasing the number of layers, the relative test errors of the FNOs deteriorates for $L > 2$, after initially decreasing. In contrast, the accuracy of the IFNOs monotonically improves for increasing values of L. Also in this case, the FNOs suffer from the vanishing gradient: the training becomes challenging when $L > 16$. In Figure 5, we depict both solutions and prediction errors obtained with the best IFNO, the best FNO, and the deepest FNO, in correspondence of two pairs of source field $g(\boldsymbol{x})$ and boundary condition $u_D(\boldsymbol{x})$. In particular, in this setting, the IFNO reaches its best test error, 0.49%, when $L = 32$. For the FNO, the best performance is achieved when $L = 2$, where the test error is 0.53%. When $L > 2$, The IFNOs consistently outperforms the FNOs in the testing experiments.

4.2. The deformation of a hyperelastic and anisotropic fiber-reinforced material

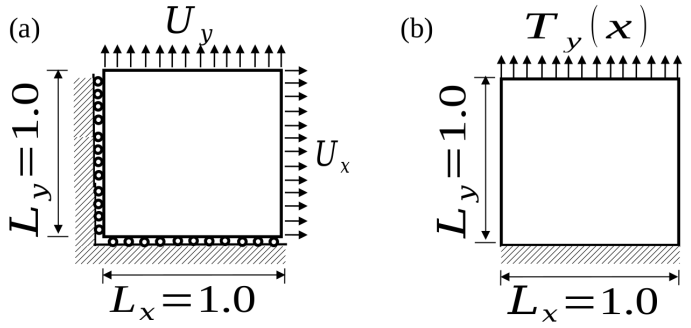

Figure 6: Problem setup of example 2: the deformation of a hyperelastic and anisotropic fiber-reinforced material. (a) A unit square subject to biaxial stretching with Dirichlet-type boundary conditions. (b) A unit square subject to uniaxial tension with Neumann-type boundary condition.

We now consider the modeling problem of a hyperelastic, anisotropic, fiber-reinforced material, and seek to find its displacement field $\boldsymbol{u} : [0,1]^2 \to \mathbb{R}^2$ under different boundary loadings. To generate training and test samples, the Holzapfel-Gasser-Odgen (HGO) model [88] was employed to describe the constitutive

behavior of the material in this example, with its strain energy density function given as:

$$\eta = \frac{c_{10}}{2}(\overline{I}_1 - 3) - c_{10}\ln(J) + \frac{k_1}{2k_2}\sum_{i=1}^{2}(\exp\left(k_2\langle E_i\rangle^2\right) - 1) + \frac{K}{2}\left(\frac{J^2-1}{2} - \ln J\right).$$

Here, $\langle\cdot\rangle$ denotes the Macaulay bracket, and the fiber strain of the two fiber groups is defined as:

$$E_i = \kappa(\overline{I}_1 - 3) + (1 - 3\kappa)(\overline{I}_{4i} - 1), \quad i = 1, 2,$$

where k_1 and k_2 are fiber modulus and the exponential coefficient, respectively, c_{10} is the moduli for the non-fibrous ground matrix, K is the bulk modulus, and κ is the fiber dispersion parameter. Moreover, $\overline{I}_1 = \mathrm{tr}(\mathbf{C})$ is the is the first invariant of the right Cauchy-Green tensor $\mathbf{C} = \mathbf{F}^T\mathbf{F}$, $\mathbf{F}$ is the deformation gradient, and J is related with $\mathbf{F}$ such that $J = \det\mathbf{F}$. For the $i-$th fiber group with angle direction α_i from the reference direction, $\overline{I}_{4i} = \mathbf{n}_i^T\mathbf{C}\mathbf{n}_i$ is the fourth invariant of the right Cauchy-Green tensor $\mathbf{C}$, where $\mathbf{n}_i = [\cos(\alpha_i), \sin(\alpha_i)]^T$. In our simulations, we considered a material with fiber reinforcement in the vertical direction, and set the orientation for both fiber groups as $\alpha_i = \pi/2$. All parameter values are summarized in Table 4.

In this example, our goal is to learn the solution operator of the HGO model, and predict the displacement field $\boldsymbol{u}(\boldsymbol{x})$ subject to different boundary conditions. As depicted in Figure 6, two types of boundary conditions are considered: (i) the Dirichlet-type boundary condition where a uniform uniaxial displacement loading was applied on the right and top edges of the plate (see Figure 6(a)); and (ii) the Neumann-type boundary loading where we applied a uniaxial tension $\boldsymbol{t}(\boldsymbol{x})$ on the top edge (see Figure 6(b)). For both cases, to generate the high-fidelity (ground-truth) dataset, we solved the displacement field on the entire domain by minimizing potential energy using the finite element method implemented in FEniCS [87]. In particular, the displacement filed was approximated by continuous piecewise linear finite elements with triangular mesh, and the grid size was taken as 0.025. Then, the finite element solution was interpolated onto χ, a structured 41×41 grid which will be employed as the discretization in our neural operators.

Learning material responses from displacement boundary conditions. We first studied the performance of the IFNOs as a solution operator under Dirichlet-type boundary conditions. To mimic the real-world mechanical test settings (see, e.g. [28]), we generated 9 different biaxial loading protocol sets as listed in Table 5, with 100 samples for each set. For each sample, a uniform uniaxial displacement boundary condition $\boldsymbol{u}_D = (U_x, 0)$ was applied on the right edge of the plate, and another uniform uniaxial displacement $\boldsymbol{u}_D = (0, U_y)$ was prescribed on the top edge. The other two edges were set as clamped on the tangential direction. Based on this boundary condition, we generated the displacement field solution $\boldsymbol{u}(\boldsymbol{x})$ using FEniCS, serving as the high-fidelity solution. Then, the neural operators were employed to learn the mapping from $\boldsymbol{f}(\boldsymbol{x}) := [\boldsymbol{x}, \tilde{U}_x, \tilde{U}_y]$ to $\boldsymbol{u}(\boldsymbol{x}) := [u_x(\boldsymbol{x}), u_y(\boldsymbol{x})]$, where $\tilde{U}_x$ and $\tilde{U}_y$ are the padded boundary conditions, as described in (2.2). Two study scenarios were considered to evaluate the in-distribution prediction capability and the out-of-

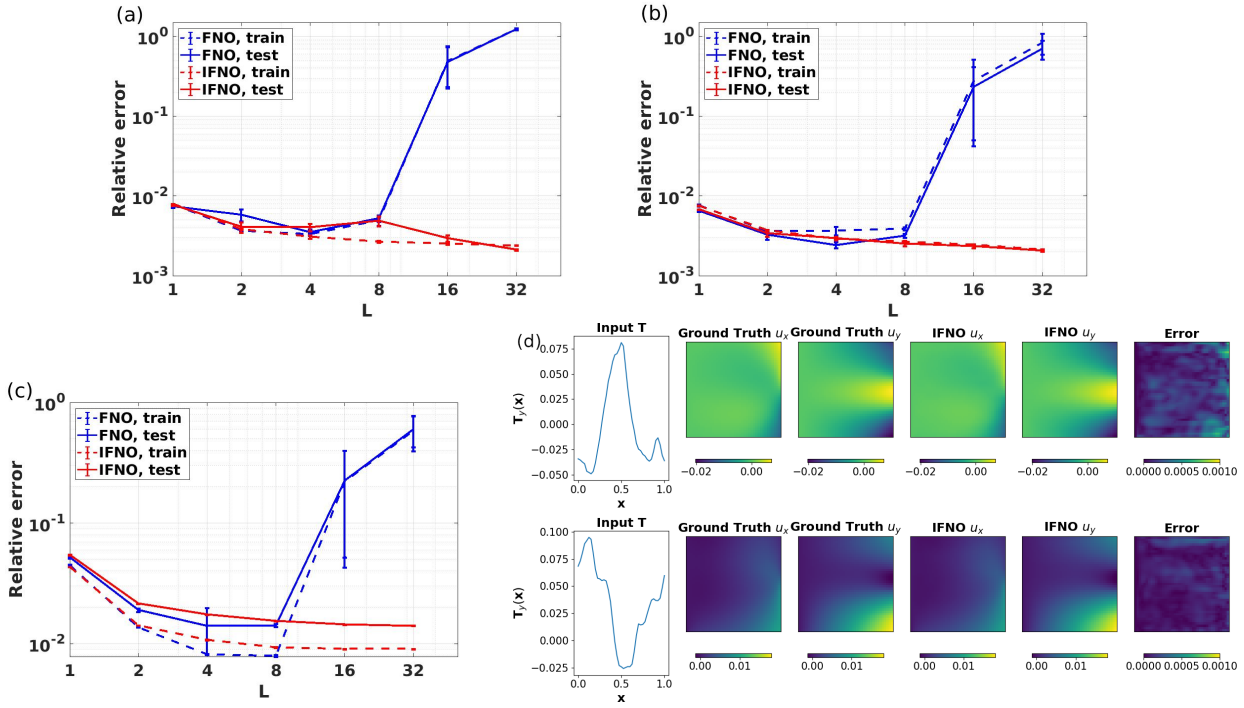

Figure 7: The deformation of a hyperelastic and anisotropic fiber-reinforced material (example 2). (a-b) Comparison of relative mean squared errors of displacement field predictions driving by displacement boundary conditions. (a) Results from in-distribution tests, where the testing boundary conditions are **inside** the training region. (b) Results from out-of-distribution tests, where the testing boundary conditions are **outside** the training region. (c-d) Results of displacement field predictions driving by traction conditions. (c) Comparison of relative mean squared errors. (d) A visualization of $L = 32$ IFNO performances on two instances of traction loads on the top edge.

distribution generalizability of the proposed neural operators:

1. We randomly selected 100 samples as the test dataset from the 900 total samples, and used all other samples to form the training dataset. In this scenario, we note that the boundary conditions of test samples are inside the training region.

2. We used protocol set #4 (0.33 : 1 biaxial tension) as the test dataset, and all other sets as the training dataset. Note that the 0.33 : 1 biaxial tension protocol is not covered in any of other sets. Therefore, with this scenario we aim to study the generalizability of the proposed method by testing with boundary conditions outside the training region.

For both study scenarios, we set the dimension of $\boldsymbol{h}$ as $d = 32$ and the number of truncated Fourier modes as $k = 8 \times 8$ in all neural operator models. For this example, we trained the network for 500 epochs with a learning rate of $5e-3$, then decreased the learning rate with a ratio of 0.5 every 100 epochs.

In Figure 7(a-b), we provide the averaged relative mean squared errors as functions of hidden layer numbers L. In Figure 7(a), we depict the results from scenario 1. We observed that when $L > 4$, both the training and testing errors from the FNOs start to increase, due to the vanishing gradient issue. A similar phenomenon is observed in Figure 7(b), where the results from scenario 2 are provided. In contrast,

the accuracy of the IFNOs monotonically improves for increasing values of L. For the in-distribution test scenario, the IFNOs reaches its lowest test error (0.21%) at $L = 32$, which almost halved the optimal error from FNOs (0.35% at $L = 4$). Similarly, for the out-of-distribution scenario, the best performance for the IFNO is obtained at $L = 32$, and the test error is 0.21%. In the mean time, the optimal FNO is still with $L = 4$, and achieved a slightly larger test error (0.24%). When comparing between the IFNO and the FNO with the same depth, the IFNO again achieves a better accuracy whenever the network is deeper than 4. Different from example 1, in this example we did not observe much overfitting problem, possibly due to the fact that the material microstructure and loading settings have low complexity: the material is assumed to be homogeneous, and the displacement-type boundary conditions are uniform. All these facts are anticipated to reduce the complexity of this learning task, so the material responses in testing datasets do not vary much from the responses in the training dataset, even in the out-of-distributing prediction scenario.

Learning material responses from traction boundary conditions. In the previous examples and experiments, we have investigated the performance of integral neural operators on predicting material responses driven by Dirichlet-type boundary conditions. Here, we further studied the material deformation driven by Neumann-type boundary conditions, as depicted in Figure 6(b). With the IFNOs, we aim to learn the solution operator which predicts the resultant displacement field $\boldsymbol{u}(\boldsymbol{x})$ driven by different traction boundary conditions $\boldsymbol{t}(\boldsymbol{x}) = [0, T_y(\boldsymbol{x})]$. In this context, the input function is $\boldsymbol{f}(\boldsymbol{x}) := [\boldsymbol{x}, \tilde{T}_y(\boldsymbol{x})]$, where $\tilde{T}_y(x, y) := T_y(x, 1)$ is the padded function of $T_y(\boldsymbol{x})$ onto the whole domain Ω. The output function is the displacement field. To generate the training/testing dataset, we sampled $1,000$ different vertical traction conditions $T_y(\boldsymbol{x})$ on the top edge from a random field, following the algorithm in [67, 89]. In particular, $T_y(\boldsymbol{x})$ is taken as the restriction of a 2D random field, $\phi(\boldsymbol{x}) = \mathcal{F}^{-1}(\gamma^{1/2}\mathcal{F}(\Gamma))(\boldsymbol{x})$, on the top edge. Here, $\Gamma(\boldsymbol{x})$ is a Gaussian white noise random field on $\mathbb{R}^2$, $\gamma = (w_1^2 + w_2^2)^{-\frac{5}{4}}$ represents a correlation function, and w_1, w_2 are the wave numbers on x and y directions, respectively. Then, for each sampled traction loading, we performed a FEniCS simulation based on the HGO model, to obtain the solutions in the entire domain and collect the corresponding solutions of displacement fields of in Ω. Among these $1,000$ samples, 800 cases were employed as the training data while the rest was kept as testing data. In this setting, we trained the network for 500 epochs with a learning rate of 5e-3, then decreased the learning rate with a ratio of 0.5 every 100 epochs.

In Figure 7(c), we show the relative mean squared errors from each neural operator model with respect to different hidden layer numbers L. Similarly to the Dirichlet-type boundary cases, the IFNO achieves its best performance at $L = 32$, while the FNO suffers from vanishing gradient when $L > 8$. In Figure 7(d), we compare the horizontal displacement u_x and vertical displacement u_y in Ω between the FEniCS ground truth and the $L = 32$ IFNO prediction, together with the prediction errors. To illustrate the network generalizability to different traction boundary conditions, the results on two instances of $T_y(\boldsymbol{x})$ among the test samples are illustrated. We observe that the IFNO predictions match well with the ground truth solution, demonstrating the capability of our proposed method in predicting material responses driven by unseen traction conditions.

4.3. The brittle fracture mechanics in glass-ceramics

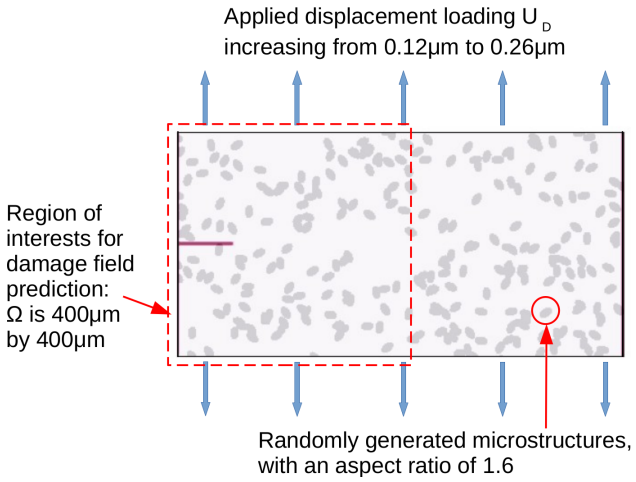

Figure 8: Problem setup of pre-cracked glass-ceramics experiment with randomly distributed material property fields and the plate microstructure considered in example 3, following [90]. Here, dark grey represents the crystalline and light grey represents the glassy matrix. This microstructure represents a glass-ceramic sample where the crystals occupy 20% of the volume.

	Young's modulus	Poisson ratio	Fracture energy	Fracture Toughness
Glass	E_1 =80 GPa	0.25	G_1 =6.59 J/m^2	0.75 MPa$\cdot\sqrt{\text{m}}$
Crystal	E_2 =133 GPa	0.25	G_2 =86.35 J/m^2	3.5 MPa$\cdot\sqrt{\text{m}}$

Table 6: Material parameters used for generating the high-fidelity solution in the pre-cracked glass-ceramics experiment, following [90].

In this example, we study the problem of brittle fracture in a glass-ceramic material, as a prototypical exemplar on the heterogeneous material damage field prediction. A glass-ceramic material is the product of controlled crystallization of a specialized glass composition, which results in the creation of a microstructure composing of one of more crystalline phases within the residual amorphous glass [91–95]. In glass-ceramics, the material has enhanced strength and toughness compared to pure glass, while the microstructure and phase assemblage of each material sample play a vital role in determining material strength and toughness.

We considered a pre-notched idealized microstructural realization which is subject to displacement boundary conditions on its top and bottom boundaries. As demonstrated in Figure 8, a plate of dimensions 800 μm by 400 μm was considered, with an initial crack of length 100 μm, and a gradually increasing uniform displacement loading U_D applied on the top and bottom of the sample. All other boundaries, including the new boundaries created by cracks, were treated as free surfaces. This microstructure realization is composed of randomly distributed crystals embedded in a glassy matrix, such that the crystals occupy 20% of the volume. Similarly to [90, 91, 95], we generated the center location (C_x, C_y) and rotation angle C_η of each crystal as random variables, satisfying $C_x \sim \mathcal{U}(0, 800)$, $C_y \sim \mathcal{U}(0, 400)$, and $C_\eta \sim \mathcal{U}(0, 2\pi)$. All crystals are identical ellipses with semi-major and semi-minor axes being 12 μm and 7.5 μm, respectively, with an aspect ratio of 1.6. The mechanical properties of glass and crystalline phases are summarized in Table 6. This material was studied experimentally in [90] and numerically in [91, 95] for different crystallized volume fractions. Here, we

adopted the setting in [95] and employed the quasi-static linear peridynamic solid (LPS) model to generate the high-fidelity simulation data. In particular, for each microstructure realization, we used $R(\boldsymbol{x})$ to denote the microstructure, such that

$$R(\boldsymbol{x}) = \begin{cases} 0 & \text{if the material point } \boldsymbol{x} \text{ is glass,} \\ 1 & \text{if the material point } \boldsymbol{x} \text{ is crystal.} \end{cases} \tag{4.1}$$

For this microstructure sample, the field of Young's modulus $E(\boldsymbol{x})$ and fracture energy $G(\boldsymbol{x})$ can be represented as linear transformations of R:

$$E(\boldsymbol{x}) = R(\boldsymbol{x})(E_2 - E_1) + E_1, \quad G(\boldsymbol{x}) = R(\boldsymbol{x})(G_2 - G_1) + G_1,$$

where E_1, E_2 are the Young's modulus of glass and crystal, respectively, and G_1, G_2 are their repsective fracture energy. The high-fidelity material responses and crack propagation simulations in this sample are calculated using the LPS model proposed in [95]:

$$\begin{aligned} &\mathcal{K}_R \boldsymbol{u}(\boldsymbol{x}, \tau) = 0, && \boldsymbol{x} \in \Omega \\ &\boldsymbol{u}(\boldsymbol{x}, \tau) = \boldsymbol{u}_D(\boldsymbol{x}, \tau), && \boldsymbol{x} \in \mathcal{BB}\Omega_D \end{aligned} \tag{4.2}$$

where $\mathcal{BB}\Omega_D$ denotes the nonlocal boundary layer on the top and the bottom edges of the plate, the instant τ denotes the indexes for (incrementally increasing) loading. In particular, we set $\boldsymbol{u}_D(\boldsymbol{x}, \tau) = [0, U_D(\tau)]$ on the top edge, and $\boldsymbol{u}_D(\boldsymbol{x}, \tau) = [0, -U_D(\tau)]$ on the bottom edge. To perform quasi-static simulations of crack propagation, we gradually increase U_D from $0.12\,\mu$m to $0.26\,\mu$m, and simulate the propagation of the crack starting from the pre-crack tip till it reaches the right boundary of the domain. At each quasi-static step, we increased U_D by $0.002\,\mu$m, performed subiterations until no new broken bonds are detected, and then proceeded to the next step. For spatial discretization, we employed uniform grids with grid size $\Delta x = 2\,\mu$m. Therefore, the whole computational domain $\Omega \cup \mathcal{BB}\Omega_D$ has $87,969$ grid points in total. To generate the training and testing samples, we employed the meshfree method proposed in [95] to solve for the displacement field $\boldsymbol{u}(\boldsymbol{x}, \tau)$ and the damage field $d(\boldsymbol{x}, \tau)$. For the detailed formulation of the LPS operator $\mathcal{K}_R$ and the numerical method, we refer interested readers to [95].

Setting and results of the glass-ceramics fracture problem. In this context, our goal is to learn the solution operator of the quasi-static LPS equation, and compute the damage field for a given plate. As shown in Figure 8, we considered a plate with a fixed microstructure field $R(\boldsymbol{x})$, and the goal is to estimate the evolution of crack in the left half of this plate, by predicting the damage field $d(\boldsymbol{x}, \tau)$ subject to increasing Dirichlet-type boundary loadings $U_D(\tau)$. That means, the neural operator were employed to learn the mapping from $\boldsymbol{f}(\boldsymbol{x}) := [\boldsymbol{x}, d(\boldsymbol{x}, \tau - \Delta\tau), \tilde{U}_D(\tau)]$ to $d(\boldsymbol{x}, \tau)$, where $d(\boldsymbol{x}, \tau - \Delta\tau)$ stands for the damage field corresponding to the last quasi-static loading step. With this setting, we aim to predict the crack propagation

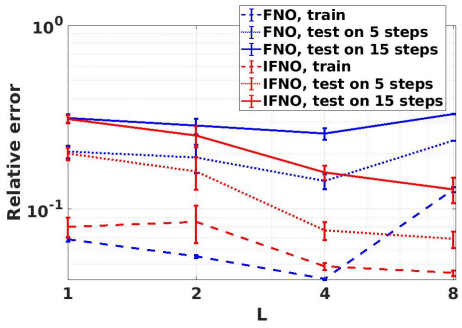

Figure 9: The glass-ceramic crack propagation problem (example 3). Comparison of relative mean squared errors for quasi-static damage field prediction on a fixed microstructure field and increasing boundary displacement loading.

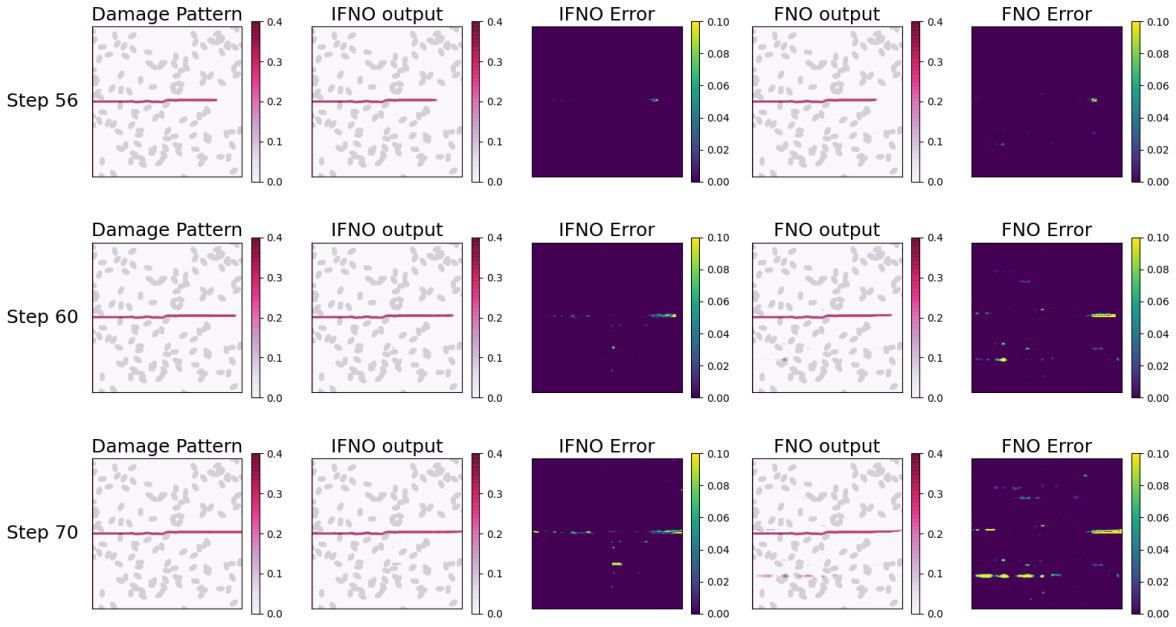

Figure 10: The glass-ceramic crack propagation problem (example 3). A visualization of FNO and IFNO performances on the 1st, 5th and 15th prediction steps. Here, the best IFNO results ($L = 8$) and best FNO results ($L = 4$) are reported.

of one particular material sample under a new and unseen loading scenario. In particular, we generate 70 numbers of samples corresponding to $U_D(\tau) \in [0.12\,\mu\text{m}, 0.26\,\mu\text{m}]$ with an increment of $\Delta U_D = 0.002\,\mu\text{m}$ for each quasi-static step, such that the first 55 steps/samples (corresponding to $U_D(\tau) \in [0.12\,\mu\text{m}, 0.23\,\mu\text{m}]$) are employed for training, and the last 15 steps/samples (corresponding to $U_D(\tau) \in [0.232\,\mu\text{m}, 0.26\,\mu\text{m}]$) are for testing. This setting reflects to the material defect monitoring scenario where some cracks are detected on a material sample with a unknown microstructure, and the learning goal is to predict and monitor the future crack growth. Therefore, this is an out-of-distribution test problem: the longer the prediction period (corresponding to larger U_D) is, the harder the prediction task will be. For the purpose of training, we choose the loss function as the accumulated error of the damage field $d(\boldsymbol{x}, \tau)$ within five successive quasi-static steps. Specifically, we used the neural operator to map $[\boldsymbol{x}, d(\boldsymbol{x}, \tau - \Delta\tau), \tilde{\boldsymbol{U}}_D(\tau)]$ to $\overline{d}(\boldsymbol{x}, \tau)$, then used $[\boldsymbol{x}, \overline{d}(\boldsymbol{x}, \tau), \tilde{\boldsymbol{U}}_D(\tau + \Delta\tau)]$ as the input to obtain $\overline{d}(\tau + \Delta\tau)$, and repeat till an approximated damage field for the next five steps are obtained. Then, we train the network by minimizing the averaged error of $\overline{d}(\boldsymbol{x}, \tau + k\Delta\tau)$, $k = 0, \cdots, 4$. A similar setting can be found, e.g., in [51]. In this example, a structured 200×200 grid is employed as the discretization in our neural operators. For both FNO and IFNO, we set the dimension of $\boldsymbol{h}$ as 48, and the truncated fourier modes $k = 20 \times 20$. For each depth L, we trained the neural network for 500 epochs with a learning rate of $5e-3$, then the learning rate was decreased by 0.5 every 100 epochs.

In Figure 9 we report the averaged relative mean squared errors as a function of iterative layer number L. In particular, we show the averaged prediction errors for a relatively short term (over 5 prediction steps) and longer term (over 15 prediction steps), respectively. When considering the short term prediction error, we observe that as we increase L from 1 to 8, the prediction error from IFNO has a monotonic and drastic decrease from 20.1% to 6.8%, reaching a similar level as the averaged training error. Hence, in this example using deeper layer is necessary to obtain a sufficiently expressive IFNO. In contrast, FNO reaches its best performance at $L = 4$, and obtains only 14.2% prediction error. For the longer term prediction error, a 12.8% averaged prediction error is obtained for IFNO at $L = 8$, while the error from the best FNO is only 25.7%. Similarly to previous examples, when increasing L the performance of FNOs starts to get polluted by the network instability issue caused by overfitting and vanishing gradient problems, which limits FNOs performance on deeper layers. In Figure 10, we show the predicted damage fields obtained with the best IFNO and the best FNO, in correspondence of the 1^{st}, 5^{th} and 15^{th} prediction steps, which are the 56^{th}, 60^{th} and 70^{th} steps among all samples. Both the solutions and the errors are plotted, showing that the FNO starts to mistakenly predict a subcrack since the 1st prediction step, and this crack grows over time, eventually leads to the large long-term prediction error in FNOs. From this example, we find that developing a stable and deep NN is important, especially for a complex learning task like the heterogeneous material damage problem.

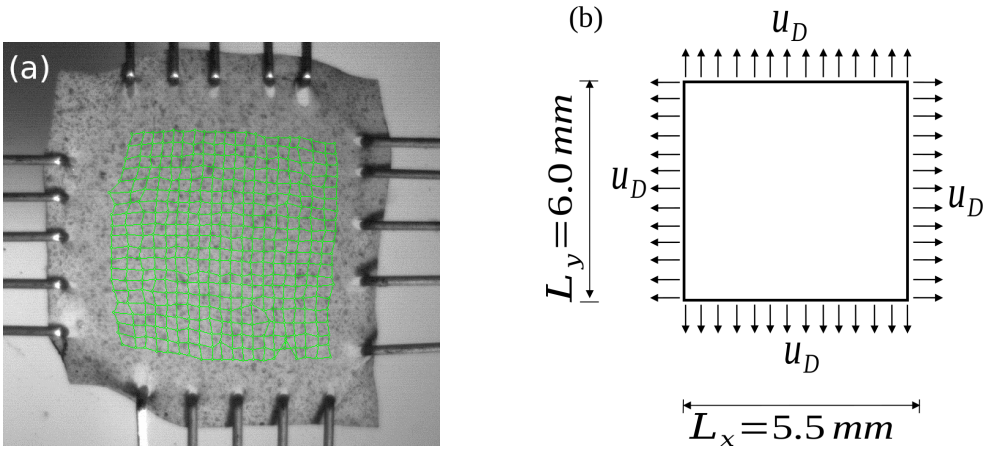

Figure 11: Problem setup of the DIC data acquisition in the latex glove sample modeling problem. (a) An image of the speckle-patterned specimen subject to biaxial stretch loading. (b) A sample subject to Dirichlet-type boundary conditions, as the corresponding numerical setting of (a).

5. Application: Learning From Digital Image Correction (DIC) Measurements

Having illustrated the performances of our learned neural operators on high-fidelity synthetic simulation datasets in Section 4, we now consider a problem of learning the material response of a latex glove sample from DIC displacement tracking measurements as a prototypical exemplar. The main objective of this section is to provide a proof-of-principle demonstration that the framework introduced thus far applies to learning tasks where the constitutive equations and material microstructure are both unknown, and the dataset has unavoidable measurement noise. Besides the FNOs, in this application we further compare our proposed IFNO against two conventional approaches that use constitutive modeling with parameter fitting to demonstrate the advantages of neural operator models and the importance of considering the heterogeneity of material microstructures.

5.1. Digital Image Correction (DIC) and biaxial mechanical testing

In this section, we first introduce the experimental sample and data acquisition procedure. For material sample acquisition, the central, palm region of a standard nitrile glove (Dealmed, NY, USA) was sectioned into a 7.5 mm $\times$7.5 mm specimen. Then, an optics-based laser thickness measurement device (Keyence, IL, USA) was used to measure the thickness of the specimen before application of a speckle pattern. Following the common procedures from previous research works [96–98], we used an airbrush to generate a random speckling texture on the surface of the specimen. Then, speckle-patterned specimens were mounted to a biaxial mechanical testing device (CellScale Biomaterials Testing Co., Canada) using five BioRake tines that pierced the specimen at each edge (Figure 11(a)). Biaxial characterizations of the specimen was conducted with 3 loading/unloading cycles, targeting an arbitrary force of 750 mN in each direction. Throughout the test, the load cell force readings and actuator positions were recorded at a frequency of 5 Hz, which were subsequently used to calculate the stresses and the stretches for the constitutive model fitting approach as

one of the baselines. Meanwhile, a CCD camera captured images throughout the biaxial test at a frequency of 5 Hz. The recorded images were tracked using the digital image correlation (DIC) module of the CellScale LabJoy software. The central 6 mm $\times$5.5 mm region of the specimen was selected for the DIC tracking, as the speckling pattern was more random and less susceptible to tracking errors. A 20×20 node grid was constructed, and the tracked coordinates were exported.

Based on the tracked coordinates, we constructed two datasets: (i) an original dataset obtained directly from the experimental measurement, and (ii) a smoothed dataset where a moving least-squares (MLS) algorithm was used to calculate the smoothed nodal displacements. To generate the displacement field $\boldsymbol{u}^{ori}(\boldsymbol{x})$ for original samples, we subtracted each material point location with its initial location on the first sample, and the boundary displacement loading was obtained by restricting $\boldsymbol{u}^{ori}(\boldsymbol{x})$ on the boundary nodes. To create a structured grid for FNOs and IFNOs, we further applied a cubic spline interpolation to the displacement field on a structured 21×21 node grid. Our goal was then to predict the displacement field in the current loading step, given the displacement on the previous step and the current boundary displacement. To construct the smoothed samples for the j^{th} material point, $\boldsymbol{x}_j = (x_j, y_j)$, we employed a two-dimensional MLS shape function Ψ_j to reconstruct the smoothed displacement field:

$$\mathbf{u}(x,y) = \sum_{j=1}^{NP} \Psi_j(x,y)\mathbf{u}_j = \sum_{j=1}^{NP} \phi(x-x_j, y-y_j; w)\mathbf{H}^T(0,0)\mathbf{M}^{-1}(x,y)\mathbf{H}(x-x_j, y-y_j)\mathbf{u}_j,$$

where $\mathbf{u}_j = [u_{xj}, u_{yj}]^T$ is the displacement vector of the j^{th} point, $\phi(x,y;w)$ is the window function with a support of w, $\mathbf{H}(x,y) = [1,x,y]^T$ is the monomial basis function of linear order, $\mathbf{M}(x,y) := \sum_{k=1}^{NP} \phi_k(x - x_k, y - y_k)\mathbf{H}(x-x_k, y-y_k)\mathbf{H}^T(x-x_k, y-y_k)$ is the moment matrix, and NP is the set of discrete points used to represent the region of interest [99, 100].

For this study, we chose $NP = 9$, a cubic B-spline function with a support of $w = 5$ for $\phi(x,y;w)$, and a 14×14 query point grid. The MLS shape functions were used to obtain the smoothed nodal displacements $\boldsymbol{u}^{sm}$. Both the smoothed and the original datasets have 877 total time instants (samples), denoted as $\mathcal{D}^{sm} = \{(\boldsymbol{u}_D)_j^{sm}, \boldsymbol{u}_j^{sm}\}_{j=1}^{877}$ and $\mathcal{D}^{un} = \{(\boldsymbol{u}_D)_j^{ori}, \boldsymbol{u}_j^{ori}\}_{j=1}^{877}$, respectively. For training and cross-validation, we randomly select 177 samples from the each dataset as test samples, and use the rest as training samples. On each dataset, these training samples were employed for parameter fitting in the constitutive modeling approaches, and used to train for the best neural operators for the IFNO and FNO. In that context, we used common datasets for the constitutive modeling approaches and neural operator learning approaches, to provide a fair comparison between the two different approaches.

5.2. Constitutive modeling for comparisons with the IFNO

In this section, we provide details for two constitutive modeling approaches, one uses constitutive model fitting to the stress-stretch data and the other uses finite element modeling of the DIC-tracked node displacements, for comparisons with the proposed IFNO method. For both approaches, a generalized Mooney-Rivlin

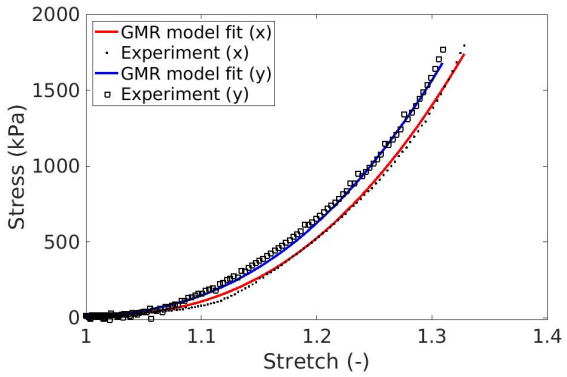

Figure 12: An illustration of the constitutive model fitting approach which optimizes the generalized Mooney Rivlin (GMR) model parameters from the stress-stretch curve for a latex glove sample.

(GMR) hyperelastic model was considered, with its strain energy density function given by:

$$\eta(\overline{I}_1, \overline{I}_2) = c_{10}(\overline{I}_1 - 3) + c_{01}(\overline{I}_2 - 3) + c_{20}(\overline{I}_1 - 3)^2 + c_{02}(\overline{I}_2 - 3)^2 + c_{11}(\overline{I}_1 - 3)(\overline{I}_2 - 3).$$

Here, $\overline{I}_1 = \mathrm{tr}(\mathbf{C})$ and $\overline{I}_2 = \frac{1}{2}[\mathrm{tr}(\mathbf{C})^2 - \mathrm{tr}(\mathbf{C}^2)]$ represent the first and second invariants of the right Cauchy-Green deformation tensor $\mathbf{C}$, and c_{ij} are the model-specific parameters. Based on this pre-assumed constitutive model, we aim to find the optimal parameters of c_{ij} from the training samples, and these parameters will then be used for displacement field predictions on the test samples.

In the first modeling approach, constitutive model parameters were obtained by fitting the final unloading portion of the biaxial stress-stretch data. In particular, the first Piola-Kirchhoff stresses in the x- and y-directions were determined using the specimen thickness t, the undeformed edge lengths L_x and L_y, and the measured forces F_x and F_y as $P_{xx} = F_x/tL_y$ and $P_{yy} = F_y/tL_x$. Meanwhile, the stretches in the two directions were calculated as the ratio of the deformed edge lengths to the undeformed length. Both stress-stretch curves in the x- and y-directions are shown in Figure 12. To obtain the optimal parameters for the GMR model, we used a differential evolution optimization framework to minimize the residual errors in stress predictions between the experimental and model predicted data. Then, using the determined model parameters, finite element modeling was performed using the DIC-tracked nodes and the relative errors of displacement fields are evaluated by comparing the result from this finite element solver and the displacement measurements from DIC. In the following contents, we will refer to this approach as the "GMR model fitting" method.

As the second modeling approach, we optimized the constitutive model parameters by minimizing the displacement error from the finite element solver directly. In particular, the structured nodal locations were imported to Abaqus [101] to construct a 21×21 node domain composed of plane stress elements. Then, we solved for the displacement field based on the GMR model using Abaqus, and calculated its relative error with respect to the experimentally-retrieved displacements of each node. The optimal model parameters

were obtained by minimizing the total relative displacement error on all training samples. In the following contents, we refer to this approach as the "GMR inverse analysis" method.

5.3. Results and discussion

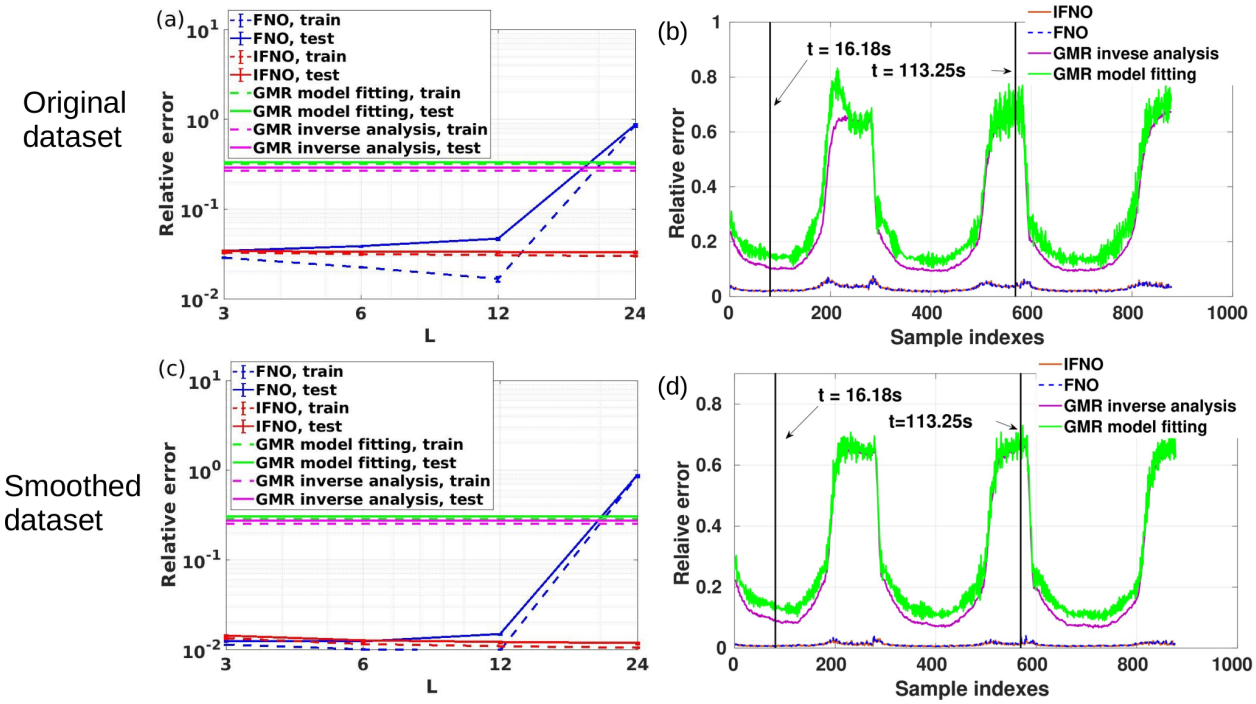

Figure 13: A latex glove sample modeling from DIC measurements. Error comparisons of each model. Upper plots: results from the original dataset. Bottom plots: results from the smoothed dataset. Left column: relative mean squared errors for quasi-static displacement field prediction on the training and test datasets. Right column: sample-wise error comparison on all samples.

In this section, we introduce the settings of our neural operator learning models and report the comparison results. Because the time instance between two subsequent loading steps is relatively long, we employed a quasi-static model. In this context, we aim to predict the displacement field $\boldsymbol{u}(\boldsymbol{x})$ based on a given boundary displacement loading $\boldsymbol{u}_D(\boldsymbol{x})$ and the displacement field from the last loading step (denoted as $\boldsymbol{u}^{last}(\boldsymbol{x})$). Therefore, the neural operators were employed to learn the mapping from $\boldsymbol{f}(\boldsymbol{x}) := [\boldsymbol{x}, \boldsymbol{u}^{last}(\boldsymbol{x}), \tilde{\boldsymbol{u}}_D(\boldsymbol{x})]$ to $\boldsymbol{u}(\boldsymbol{x})$, where $\tilde{\boldsymbol{u}}_D$ is the zero-padded boundary condition, as described in (2.2). In this example, for both the FNOs and IFNOs, we set the dimension of $\boldsymbol{h}$ as $d = 16$, and the number of truncated Fourier modes as $k = 8 \times 8$. For each depth L, we train the neural network for 1,000 epochs with a learning rate of $1e-3$, then decrease the learning rate with a ratio of 0.7 every 100 epochs.

In Figure 13, we report the relative mean squared errors from both the original dataset (see Figure 13(a)) and the smoothed dataset (see Figure 13(c)), as functions of the number of hidden layers L from 3 to 24. The sample-wise error for each model are also provided in Figure 13(b) for the original dataset and in Figure 13(d) for the smoothed dataset. Unsurprisingly, when comparing the results from the original dataset and the smoothed dataset, one can observe that the smoothing procedure improves the prediction accuracy for

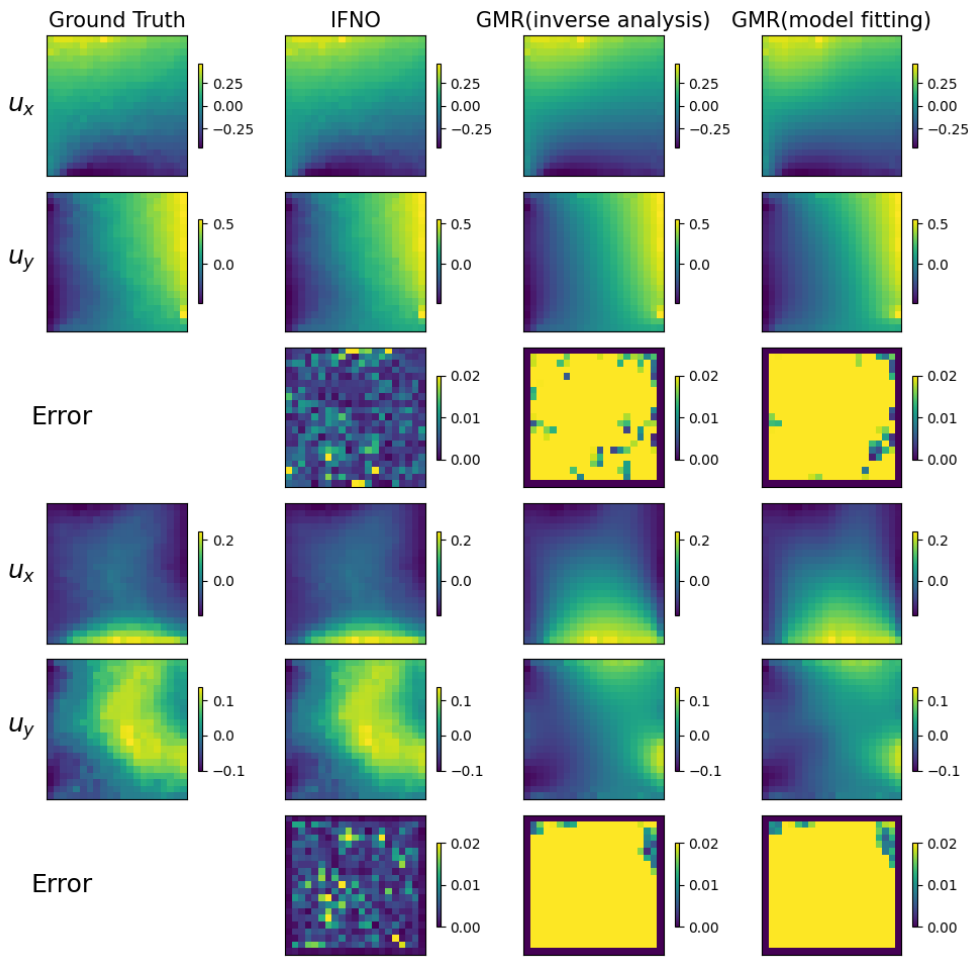

Figure 14: A latex glove sample modeling from DIC measurements. A visualization of GMR and IFNO performances on a test sample in the **original** dataset.

all models. That is because the DIC measurements may contain noise-induced errors, and the nonlocal smoothing procedure we employed performs as an effective filter [102] for the measurement noise. When comparing the prediction accuracy from different models, similar to the previous examples, the FNO suffers from overfitting and vanishing gradient issues when $L > 2$, especially in the original (more noisy) dataset. This finding is consistent with the results reported in [68, 103], where the performance of the FNOs was found to be deteriorated on noisy datasets. In contrast, the accuracy of the IFNOs monotonically improves with the increase of L. Both neural operator models outperforms the conventional constitutive modeling approaches by around one order of magnitude. Among all the models, the deep IFNO ($L = 24$) performs the best in both datasets. On the original dataset which features noise, it achieves a 3.3% prediction error. On the smoothed dataset, the IFNO has an 1.18% prediction error. On the other hand, the GMR model fitting and GMR inverse analysis approaches have obtained 33.0% and 29.1% prediction errors on the original dataset, respectively. On the smoothed dataset, the prediction error for these two GMR models are slightly smaller, as 30.5% and 27.3%, respectively. To provide further insights into this comparison, in Figures 14-15

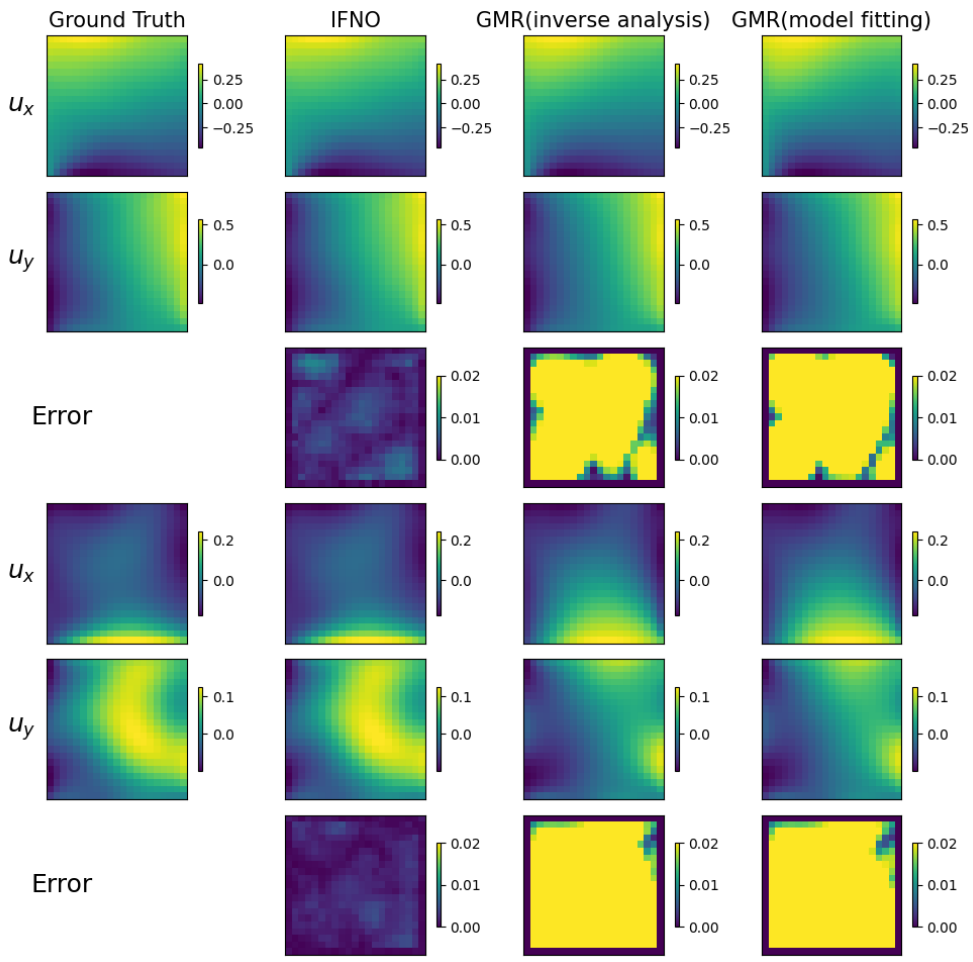

Figure 15: A latex glove sample modeling from DIC measurements. A visualization of GMR and IFNO performances on a test sample in the **smoothed** dataset.

we depict both solutions and prediction errors obtained with the best IFNO and the two GMR models on two test samples which correspond to the large deformation ($t = 113.25\,s$) and small deformation ($t = 16.18\,s$) representatives, respectively. From the ground-truth data pattern of $u_y(\boldsymbol{x})$, we can see that the glove sample is in fact heterogeneous, since a large deformation region is observed in the middle of the sample. Both GMR models fail to capture the material heterogeneity and hence obtained large prediction errors. This observation again confirms the importance of capturing the material heterogeneity and verifies the capability of IFNOs in heterogeneous material modeling.

6. Conclusion

With the objective of predicting material responses under unseen loading conditions, in this work we have proposed a novel data-driven computing paradigm for material modeling, which integrates material identification, modeling procedures, and material responses prediction into one unified learning framework. In particular, a data-driven model has been developed, which learns the mapping from loading conditions to

the corresponding material responses as a solution operator. To this end, a new integral neural operator has been proposed, which we refer to as the implicit Fourier neural operator (IFNO). In the IFNO, the increment between layers are modeled by integral operators, so the resultant architecture can be interpreted as a fixed point method for the unknown governing laws. Furthermore, by identifying its layers with time instants, the IFNO can be reinterpreted as time-dependent equations, which enables the use of efficient initialization techniques that enhances the network stability in the deep layer limit. Our results have shown that, in all learning tasks, the IFNOs outperform baseline methods in stability and prediction accuracy for unseen loading conditions. Both the universal approximation theorem and numerical results demonstrate that, in complex learning tasks, a stable deep layer architecture is necessary to achieve a satisfactory prediction accuracy. Last but not least, we have, *for the first time*, leveraged the application of neural operators to learning the material responses directly from DIC displacement tracking measurements, where the constitutive equations and material microstructure are both unknown, and measurement noise is present. Numerical results have confirmed the advantage of neural operator learning approaches against the conventional constitutive modeling approaches: the former does not require a pre-assumed material model, and is able to capture the material heterogeneity. Hence, the proposed neural operator models have outperformed the conventional generalized Mooney Rivlin (GMR) model in prediction accuracy by at least one order of magnitude. When comparing with another neural operator model, i.e., the FNOs, our proposed IFNOs have been shown to be less prone to the overfitting issue and hence achieve a better performance on noisy experimental datasets.

Although the IFNO requires a much smaller number of trainable parameters comparing with its counterpart, FNOs, and a shorter training time for each epoch, we did not observe a decrease of the computational time because the fixed point procedure of the IFNO comes with the price of using an iterative algorithm. Therefore, an important next step is to combine the IFNO with faster training techniques of implicit networks [73] to improve its efficiency. Another limitation of the current work comes from the fact that it focuses on problems with structured grids only. An interesting future direction is to extend the IFNO to handle unstructured grids, using the techniques such as dgFNO+ [68]. Moreover, we point out that the IFNO provides a general and flexible solution operator for unknown governing laws, which is not restricted to material modeling tasks. As another natural extension, we will consider the application of the IFNO on other complex learning tasks, such as image classification problems. The implicit neural operator architecture we proposed here can also be combined with other recent integral neural operator architectures, e.g., the multiwavelet-based operator [54] and the integral autoencoder-based network (IAE-Net) [53], which would be another interesting future direction.

Acknowledgements

The authors would like to thank Mr. Minglang Yin for sharing his FEniCS codes and for the helpful discussions. H. You and Y. Yu would like to acknowledge support by the National Science Foundation under award DMS 1753031. Portions of this research were conducted on Lehigh University's Research

Computing infrastructure partially supported by NSF Award 2019035. We also thank the Presbyterian Health Foundation Team Science Grant, and the National Science Foundation Graduate Research Fellowship Program (GRF2020307284).

Appendix A. Detailed Numeric Results

In this section we provide the detailed numerical results of each task in Sections 4-5, as the supplementary results of the training and test errors plotted in Figures 3, 7, 9 and 13 of the main text. The full results for porous medium pressure field learning I, porous medium pressure field learning II, fiber-reinforced material displacement field learning, glass-ceramics damage field learning, and DIC measurements of latex glove displacement filed learning are provided in Tables A1, A2, A3, A4 and A5, respectively. To reduce the impact of initialization in neural operator models, for each task we run five simulations for each network using different random seeds, and report the mean and the standard error among these five simulations. For each model, we use the bold case to highlight the architecture with the best prediction accuracy.

Model/dataset		$L = 1$	$L = 2$	$L = 4$	$L = 8$	$L = 16$	$L = 32$
IFNO	train	1.67e-2±1.16e-4	7.79e-3±5.58e-5	6.48e-3±6.16e-5	5.84e-3±6.58e-5	5.46e-3±6.79e-5	5.21e-3±6.98e-5
	test	1.77e-2±1.18e-4	1.23e-2±9.46e-5	1.10e-2±6.90e-5	1.05e-2±5.72e-5	1.04e-2±3.72e-5	**1.02e-2±5.77e-5**
FNO	train	1.65e-2±4.94e-5	4.13e-3±3.16e-4	7.94e-3±1.65e-5	6.83e-4±5.09e-6	8.34e-4±1.55e-5	2.84e-1±2.57e-6
	test	1.76e-2±9.40e-5	1.30e-2±5.11e-5	**1.19e-2±8.98e-5**	1.56e-2±1.85e-4	2.80e-2±1.19e-3	2.90e-1±1.07e-4

Table A1: Numerical results for the learning task of porous medium I. Bold numbers highlight the case with the best error for each model.

Model/dataset		$L = 1$	$L = 2$	$L = 4$	$L = 8$	$L = 16$	$L = 32$	$L = 64$
IFNO	train	9.81e-3±9.90e-5	4.38e-3±9.30e-5	3.93e-3±7.19e-5	3.89e-3±7.60e-5	3.90e-3±8.36e-5	3.98e-3±9.57e-5	3.93e-3±1.01e-4
	test	1.10e-2±1.16e-4	5.75e-3±1.17e-4	5.23e-3±1.05e-4	5.10e-3±1.16e-4	5.07e-3±1.51e-4	5.04e-3±1.61e-4	**4.89e-3±2.22e-4**
FNO	train	1.00e-2±9.52e-5	3.43e-3±8.54e-5	3.27e-3±8.53e-5	3.77e-3±2.65e-5	3.91e-3±2.54e-5	9.86e-1±2.20e-5	9.86e-1±2.19e-5
	test	1.13e-2±1.05e-4	**5.26e-3±8.21e-5**	6.07e-3±1.68e-4	8.59e-3±5.14e-5	1.26e-2±3.26e-4	9.89e-1±2.03e-4	9.89e-1±2.03e-4

Table A2: Numerical results for the learning task of porous medium II. Bold numbers highlight the case with the best error for each model.

References

[1] T. Zohdi, D. Steigmann, The toughening effect of microscopic filament misalignment on macroscopic ballistic fabric response, International journal of fracture 118 (4) (2002) 71–76.

[2] P. Wriggers, G. Zavarise, T. Zohdi, A computational study of interfacial debonding damage in fibrous composite materials, Computational Materials Science 12 (1) (1998) 39–56.

[3] Y. Kok, X. P. Tan, P. Wang, M. Nai, N. H. Loh, E. Liu, S. B. Tor, Anisotropy and heterogeneity of microstructure and mechanical properties in metal additive manufacturing: A critical review, Materials & Design 139 (2018) 565–586.

Dirichlet boundary condition with in-distribution test							
Model/dataset		$L = 1$	$L = 2$	$L = 4$	$L = 8$	$L = 16$	$L = 32$
IFNO	train	7.96e-3±6.60e-5	3.82e-3±6.10e-5	3.07e-3±1.36e-4	2.68e-3±6.40e-5	2.52e-3±4.91e-5	2.40e-3±2.42e-5
	test	7.70e-3±1.66e-4	4.10e-3±6.17e-4	4.05e-3±4.25e-4	4.86e-3±7.12e-4	2.96e-3±2.78e-4	**2.12e-3±4.42e-5**
FNO	train	7.81e-3±8.77e-5	3.65e-3±8.59e-5	3.34e-3±4.58e-5	4.98e-3±7.51e-4	4.93e-1±2.60e-1	1.23e0±2.73e-3
	test	7.46e-3±3.79e-4	5.78e-3±9.90e-4	**3.50e-3±3.26e-4**	5.24e-3±4.70e-4	4.78e-1±2.56e-1	1.24e0±3.67e-2
Dirichlet boundary condition with out-of-distribution test							
Model/dataset		$L = 1$	$L = 2$	$L = 4$	$L = 8$	$L = 16$	$L = 32$
IFNO	train	7.58e-3±8.20e-5	3.74e-3±6.09e-5	2.92e-3±6.52e-5	2.69e-3±7.10e-5	2.44e-3±3.88e-5	2.14e-3±2.33e-5
	test	6.78e-3±7.18e-5	3.42e-3±3.44e-4	2.95e-3±1.60e-4	2.51e-3±1.74e-4	2.34e-3±1.23e-4	**2.06e-3±5.92e-5**
FNO	train	7.61e-3±2.07e-4	3.63e-3±1.13e-4	3.68e-3±4.46e-4	3.89e-3±1.77e-4	2.78e-1±2.36e-1	8.34e-1±2.44e-1
	test	6.56e-3±2.62e-4	3.25e-3±4.33e-4	**2.43e-3±2.09e-4**	3.18e-3±1.98e-4	2.33e-1±1.83e-1	7.03e-1±1.85e-1
Neumann boundary condition							
Model/dataset		$L = 1$	$L = 2$	$L = 4$	$L = 8$	$L = 16$	$L = 32$
IFNO	train	4.33e-2±1.26e-4	1.42e-2±2.40e-5	1.08e-2±8.91e-5	9.32e-3±7.34e-5	9.03e-3±4.48e-5	9.04e-3±7.01e-5
	test	5.45e-2±6.35e-4	2.15e-2±1.44e-4	1.75e-2±8.42e-5	1.54e-2±8.60e-5	1.44e-2±1.30e-4	**1.41e-2±3.70e-5**
FNO	train	4.47e-2±5.61e-4	1.37e-2±4.88e-5	8.18e-3±7.51e-5	7.96e-3±1.36e-4	2.21e-1±1.78e-1	5.81e-1±1.87e-1
	test	5.19e-2±8.17e-4	1.90e-2±6.55e-4	**1.40e-2±5.80e-3**	1.42e-2±3.86e-4	2.26e-1±1.74e-1	6.01e-1±1.76e-1

Table A3: Numerical results for the learning task of fiber-reinforced material displacement field. Bold numbers highlight the case with the best error for each model.

Model/dataset		$L = 1$	$L = 2$	$L = 4$	$L = 8$
IFNO	train	8.01e-2±9.96e-3	8.48e-2±1.96e-2	4.87e-2±2.05e-3	4.48e-2±1.62e-3
	5-step test	2.01e-1±1.58e-2	1.60e-1±3.25e-2	7.65e-2±8.46e-3	**6.86e-2±7.08e-3**
	15-step test	3.10e-1±1.54e-2	2.51e-1±3.28e-2	1.58e-1±1.42e-2	**1.28e-1±2.04e-2**
FNO	train	6.85e-2±1.85e-3	5.52e-2±9.74e-4	4.16e-2±3.33e-4	1.28e-1±4.11e-3
	5-step test	2.06e-1±1.64e-2	1.91e-1±3.41e-2	**1.42e-1±1.34e-2**	2.36e-1±3.68e-4
	15-step test	3.12e-1±1.69e-2	2.84e-1±2.72e-2	**2.57e-1±1.85e-2**	3.30e-1±1.74e-3

Table A4: Numerical results for the learning task of glass-ceramics damage field. Bold numbers highlight the case with the best error for each model.

Model/dataset		$L = 3$	$L = 6$	$L = 12$	$L = 24$
IFNO, original	train	3.26e-2±1.08e-4	3.13e-2±1.30e-4	3.06e-2±1.08e-4	3.00e-2±1.24e-4
	test	3.43e-2±4.96e-4	3.34e-2±4.53e-4	3.32e-2±4.41e-4	**3.30e-2±4.63e-4**
FNO, original	train	2.88e-2±1.23e-4	2.25e-2±8.68e-5	1.66e-2±9.94e-4	8.47e-1±4.72e-3
	test	**3.40e-2±4.09e-4**	3.84e-2±4.21e-4	4.66e-2±1.47e-3	8.61e-1±2.70e-2
GMR model fitting, original	train	3.16e-1			
	test	**3.30e-1**			
GMR inverse analysis, original	train	2.66e-1			
	test	**2.91e-1**			
IFNO, smoothed	train	1.33e-2±1.61e-4	1.16e-2±8.10e-5	1.09e-2±4.91e-5	1.05e-2±6.01e-5
	test	1.43e-2±2.99e-4	1.26e-2±2.20e-4	1.21e-2±2.28e-4	**1.18e-2±2.21e-4**
FNO, smoothed	train	1.14e-2±3.28e-5	1.01e-2±9.28e-5	9.83e-3±3.02e-4	8.49e-1±3.50e-3
	test	1.25e-2±2.25e-4	**1.23e-2±1.98e-4**	1.49e-2±1.15e-4	8.73e-1±1.87e-2
GMR model fitting, smoothed	train	2.87e-1			
	test	**3.05e-1**			
GMR inverse analysis, smoothed	train	2.52e-1			
	test	**2.73e-1**			

Table A5: Numerical results for the learning task of DIC measurements of latex glove displacement filed, compared with the generalized Mooney-Rivlin (GMR) model. Bold numbers highlight the case with the best error for each model.

[4] R. Bostanabad, Y. Zhang, X. Li, T. Kearney, L. C. Brinson, D. W. Apley, W. K. Liu, W. Chen, Computational microstructure characterization and reconstruction: Review of the state-of-the-art techniques, Progress in Materials Science 95 (2018) 1–41.

[5] Z. Su, L. Ye, Y. Lu, Guided lamb waves for identification of damage in composite structures: A review, Journal of sound and vibration 295 (3-5) (2006) 753–780.

[6] 2014 technical strategic plan, Tech. rep., the Air Force Office of Scientific Research (2014).

[7] R. Talreja, J. Varna, Modeling damage, fatigue and failure of composite materials, Elsevier, 2015.

[8] J. Sorić, P. Wriggers, O. Allix, Multiscale modeling of heterogeneous structures, Springer, 2018.

[9] G. Pijaudier-Cabot, F. Dufour, Damage mechanics of cementitious materials and structures, John Wiley & Sons, 2013.

[10] C. Mourlas, G. Markou, M. Papadrakakis, Accurate and computationally efficient nonlinear static and dynamic analysis of reinforced concrete structures considering damage factors, Engineering Structures 178 (2019) 258–285.

[11] G. Markou, R. Garcia, C. Mourlas, M. Guadagnini, K. Pilakoutas, M. Papadrakakis, A new damage factor for seismic assessment of deficient bare and frp-retrofitted rc structures, Engineering Structures 248 (2021) 113152.

[12] E. A. Lindgren, US Air Force perspective on validated NDE–past, present, and future, in: AIP Conference Proceedings, Vol. 1706, AIP Publishing LLC, 2016, p. 020002.

[13] E. Lindgren, J. Brausch, C. Buynak, P. Kobryn, M. Leonard, The state of nondestructive evaluation and structural health monitoring, in: Aircraft Structural Integrity Program Conference, 2013.

[14] M. HDBK, Nondestructive evaluation system reliability assessment, Department of Defense Handbook 7.

[15] J. D. Achenbach, Quantitative nondestructive evaluation, International Journal of Solids and Structures 37 (1-2) (2000) 13–27.

[16] K. Jones, J. Brausch, W. Fong, B. Harris, Probing the future: Better f-16 inspections using conformal eddy current inspection tools, in: Proceedings of 2015 Aircraft Airworthiness & Sustainment Conference, Baltimore, Maryland, 2015.

[17] B. Pan, L. Yu, Q. Zhang, Review of single-camera stereo-digital image correlation techniques for full-field 3d shape and deformation measurement, Science China Technological Sciences 61 (1) (2018) 2–20.

[18] K. Shukla, P. C. Di Leoni, J. Blackshire, D. Sparkman, G. E. Karniadakis, Physics-informed neural network for ultrasound nondestructive quantification of surface breaking cracks, arXiv preprint arXiv:2005.03596.

[19] M. Misfeld, H.-H. Sievers, Heart valve macro-and microstructure, Philosophical Transactions of the Royal Society B: Biological Sciences 362 (1484) (2007) 1421–1436.

[20] J. Rieppo, J. Hallikainen, J. S. Jurvelin, I. Kiviranta, H. J. Helminen, M. M. Hyttinen, Practical considerations in the use of polarized light microscopy in the analysis of the collagen network in articular cartilage, Microscopy research and technique 71 (4) (2008) 279–287.

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

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

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

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

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

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

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

[28] Q. He, D. W. Laurence, C.-H. Lee, J.-S. Chen, Manifold learning based data-driven modeling for soft biological tissues, Journal of Biomechanics 117 (2021) 110124.

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

[30] R. Ibañez, D. Borzacchiello, J. V. Aguado, E. Abisset-Chavanne, E. Cueto, P. Ladeveze, F. Chinesta, Data-driven non-linear elasticity: constitutive manifold construction and problem discretization, Computational Mechanics 60 (5) (2017) 813–826.

[31] R. Ibanez, E. Abisset-Chavanne, J. V. Aguado, D. Gonzalez, E. Cueto, F. Chinesta, A manifold learning approach to data-driven computational elasticity and inelasticity, Archives of Computational Methods in Engineering 25 (1) (2018) 47–57.

[32] L. Stainier, A. Leygue, M. Ortiz, Model-free data-driven methods in mechanics: material data identification and solvers, Computational Mechanics 64 (2) (2019) 381–393.

[33] T. Kirchdoerfer, M. Ortiz, Data-driven computational mechanics, Computer Methods in Applied Mechanics and Engineering 304 (2016) 81–101.

[34] Y. Heider, K. Wang, W. Sun, So (3)-invariance of informed-graph-based deep neural network for anisotropic elastoplastic materials, Computer Methods in Applied Mechanics and Engineering 363 (2020) 112875.

[35] J. N. Fuhg, N. Bouklas, On physics-informed data-driven isotropic and anisotropic constitutive models through probabilistic machine learning and space-filling sampling, arXiv preprint arXiv:2109.11028.

[36] K. Wang, W. Sun, A multiscale multi-permeability poroplasticity model linked by recursive homogenizations and deep learning, Computer Methods in Applied Mechanics and Engineering 334 (2018) 337–380.

[37] Q. He, D. Barajas-Solano, G. Tartakovsky, A. M. Tartakovsky, Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport, Advances in Water Resources 141 (2020) 103610.

[38] A. M. Tartakovsky, C. O. Marrero, P. Perdikaris, G. D. Tartakovsky, D. Barajas-Solano, Physics-informed deep neural networks for learning parameters and constitutive relationships in subsurface flow problems, Water Resources Research 56 (5) (2020) e2019WR026731.

[39] Z. Liu, C. Wu, M. Koishi, A deep material network for multiscale topology learning and accelerated nonlinear modeling of heterogeneous materials, Computer Methods in Applied Mechanics and Engineering 345 (2019) 1138–1168.

[40] H. Yang, X. Guo, S. Tang, W. K. Liu, Derivation of heterogeneous material laws via data-driven principal component expansions, Computational Mechanics 64 (2) (2019) 365–379.

[41] K. Garbrecht, M. Aguilo, A. Sanderson, A. Rollett, R. M. Kirby, J. Hochhalter, Interpretable machine learning for texture-dependent constitutive models with automatic code generation for topological optimization, Integrating Materials and Manufacturing Innovation 10 (3) (2021) 373–392.

[42] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707.

[43] J. Bongard, H. Lipson, Automated reverse engineering of nonlinear dynamical systems, Proceedings of the National Academy of Sciences 104 (24) (2007) 9943–9948.

[44] M. Schmidt, H. Lipson, Distilling free-form natural laws from experimental data, science 324 (5923) (2009) 81–85.

[45] S.-M. Udrescu, M. Tegmark, Ai feynman: A physics-inspired method for symbolic regression, Science Advances 6 (16) (2020) eaay2631.

[46] G. Bomarito, T. Townsend, K. Stewart, K. Esham, J. Emery, J. Hochhalter, Development of interpretable, data-driven plasticity models with symbolic regression, Computers & Structures 252 (2021) 106557.

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

[48] L. Lu, P. Jin, G. Pang, Z. Zhang, G. E. Karniadakis, Learning nonlinear operators via deeponet based on the universal approximation theorem of operators, Nature Machine Intelligence 3 (3) (2021) 218–229.

[49] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Neural operator: Graph kernel network for partial differential equations, arXiv preprint arXiv:2003.03485.

[50] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, A. Stuart, K. Bhattacharya, A. Anandkumar, Multipole graph neural operator for parametric partial differential equations, Advances in Neural Information Processing Systems 33.

[51] Z. Li, N. B. Kovachki, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, A. Anandkumar, et al., Fourier neural operator for parametric partial differential equations, in: International Conference on Learning Representations, 2020.

[52] H. You, Y. Yu, M. D'Elia, T. Gao, S. Silling, Nonlocal kernel network (NKN): a stable and resolution-independent deep neural network, arXiv preprint arXiv:2201.02217.

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

[54] G. Gupta, X. Xiao, P. Bogdan, Multiwavelet-based operator learning for differential equations, in: A. Beygelzimer, Y. Dauphin, P. Liang, J. W. Vaughan (Eds.), Advances in Neural Information Processing Systems, 2021.
URL https://openreview.net/forum?id=LZDiWaC9CGL

[55] S. Goswami, M. Yin, Y. Yu, G. E. Karniadakis, A physics-informed variational deeponet for predicting crack path in quasi-brittle materials, Computer Methods in Applied Mechanics and Engineering 391 (2022) 114587.

[56] X. Guo, W. Li, F. Iorio, Convolutional neural networks for steady flow approximation, in: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 481–490.

[57] Y. Zhu, N. Zabaras, Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification, Journal of Computational Physics 366 (2018) 415–447.

[58] J. Adler, O. Öktem, Solving ill-posed inverse problems using iterative deep neural networks, Inverse Problems 33 (12) (2017) 124007.

[59] S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, S. Kaushik, Prediction of aerodynamic flow fields using convolutional neural networks, Computational Mechanics 64 (2) (2019) 525–545.

[60] Y. Khoo, J. Lu, L. Ying, Solving parametric pde problems with artificial neural networks, European Journal of Applied Mathematics 32 (3) (2021) 421–435.

[61] J. C. De los Reyes, Numerical PDE-constrained optimization, Springer, 2015.

[62] E. Weinan, B. Yu, The deep ritz method: A deep learning-based numerical algorithm for solving variational problems, Communications in Mathematics and Statistics 6 (1).

[63] L. Bar, N. Sochen, Unsupervised deep learning algorithm for pde-based forward and inverse problems, arXiv preprint arXiv:1904.05417.

[64] J. D. Smith, K. Azizzadenesheli, Z. E. Ross, Eikonet: Solving the eikonal equation with deep neural networks, IEEE Transactions on Geoscience and Remote Sensing.

[65] S. Pan, K. Duraisamy, Physics-informed probabilistic learning of linear embeddings of nonlinear dynamics with guaranteed stability, SIAM Journal on Applied Dynamical Systems 19 (1) (2020) 480–509.

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

[67] M. Yin, E. Zhang, Y. Yu, G. E. Karniadakis, Interfacing finite elements with deep neural operators for fast multiscale modeling of mechanics problems (2022). arXiv:2203.00003.

[68] L. Lu, X. Meng, S. Cai, Z. Mao, S. Goswami, Z. Zhang, G. E. Karniadakis, A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data, arXiv preprint arXiv:2111.05512.

[69] L. El Ghaoui, F. Gu, B. Travacca, A. Askari, A. Tsai, Implicit deep learning, SIAM Journal on Mathematics of Data Science 3 (3) (2021) 930–958.

[70] S. Bai, J. Z. Kolter, V. Koltun, Deep equilibrium models, in: Proceedings of the 33rd International Conference on Neural Information Processing Systems, 2019, pp. 690–701.

[71] E. Winston, J. Z. Kolter, Monotone operator equilibrium networks, Advances in Neural Information Processing Systems 33 (2020) 10718–10728.

[72] S. Bai, V. Koltun, J. Z. Kolter, Multiscale deep equilibrium models, Advances in Neural Information Processing Systems 33.

[73] S. W. Fung, H. Heaton, Q. Li, D. McKenzie, S. Osher, W. Yin, Jfb: Jacobian-free backpropagation for implicit networks, arXiv preprint arXiv:2103.12803.

[74] E. Haber, L. Ruthotto, E. Holtham, S.-H. Jun, Learning across scales—multiscale methods for convolution neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018.

[75] J. Modersitzki, FAIR: flexible algorithms for image registration, SIAM, 2009.

[76] R. J. LeVeque, Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems, SIAM, 2007.

[77] O. C. Zienkiewicz, R. L. Taylor, P. Nithiarasu, J. Zhu, The finite element method, Vol. 3, McGraw-hill London, 1977.

[78] G. Karniadakis, S. Sherwin, Spectral/hp element methods for computational fluid dynamics, OUP Oxford, 2005.

[79] M. Kim, N. Winovich, G. Lin, W. Jeong, Peri-net: Analysis of crack patterns using deep neural networks, Journal of Peridynamics and Nonlocal Modeling 1 (2) (2019) 131–142.

[80] H. You, Y. Yu, S. Silling, M. D'Elia, A data-driven peridynamic continuum model for upscaling molecular dynamics, Computer Methods in Applied Mechanics and Engineering 389 (2022) 114400.

[81] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, A. Anandkumar, Neural operator: Learning maps between function spaces, arXiv preprint arXiv:2108.08481.

[82] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, IEEE Conference on Computer Vision and Pattern Recognition.

[83] L. Ruthotto, E. Haber, Deep neural networks motivated by partial differential equations, Journal of Mathematical Imaging and Vision (2019) 1–13.

[84] N. Kovachki, S. Lanthaler, S. Mishra, On universal approximation and error bounds for fourier neural operators, Journal of Machine Learning Research 22 (2021) Art–No.

[85] A. Pinkus, Approximation theory of the mlp model in neural networks, Acta numerica 8 (1999) 143–195.

[86] S. Hochreiter, The vanishing gradient problem during learning recurrent neural nets and problem solutions, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6 (02) (1998) 107–116.

[87] M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, G. N. Wells, The fenics project version 1.5, Archive of Numerical Software 3 (100).

[88] G. A. Holzapfel, T. C. Gasser, R. W. Ogden, A new constitutive framework for arterial wall mechanics and a comparative study of material models, Journal of Elasticity and the Physical Science of Solids 61 (1) (2000) 1–48.

[89] A. Lang, J. Potthoff, Fast simulation of gaussian random fields, Monte Carlo Methods and Applications 17 (3) (2011) 195–214. doi:doi:10.1515/mcma.2011.009.
URL https://doi.org/10.1515/mcma.2011.009

[90] F. Serbena, I. Mathias, C. Foerster, E. Zanotto, Crystallization toughening of a model glass-ceramic, Acta Materialia 86 (2015) 216–228.

[91] N. Prakash, B. Deng, R. J. Stewart, C. M. Smith, J. T. Harris, Investigation of microscale fracture mechanisms in glass-ceramics using peridynamics simulations, Journal of American Ceramic Society.

[92] F. C. Serbena, E. D. Zanotto, Internal residual stresses in glass-ceramics: A review, Journal of Non-Crystalline Solids 358 (6-7) (2012) 975–984.

[93] W. Holand, G. H. Beall, Glass-ceramic technology, John Wiley & Sons, 2019.

[94] Q. Fu, G. H. Beall, C. M. Smith, Nature-inspired design of strong, tough glass-ceramics, MRS Bulletin 42 (3) (2017) 220–225.

[95] Y. Fan, H. You, X. Tian, X. Yang, X. Li, N. Prakash, Y. Yu, A meshfree peridynamic model for brittle fracture in randomly heterogeneous materials, arXiv preprint arXiv:2202.06578.

[96] D. S. Zhang, D. D. Arola, Applications of digital image correlation to biological tissues, Journal of Biomedical Optics 9 (4) (2004) 691–699.

[97] G. Lionello, L. Cristofolini, A practical approach to optimizing the preparation of speckle patterns for digital-image correlation, Measurement Science and Technology 25 (10) (2014) 107001.

[98] M. Palanca, G. Tozzi, L. Cristofolini, The use of digital image correlation in the biomechanical area: a review, International Biomechanics 3 (1) (2016) 1–21.

[99] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, P. Krysl, Meshless methods: An overview and recent developments, Computer Methods in Applied Mechanics and Engineering 139 (1-4) (1996) 3–47.

[100] J.-S. Chen, C. Pan, C.-T. Wu, W. K. Liu, Reproducing kernel particle methods for large deformation analysis of non-linear structures, Computer Methods in Applied Mechanics and Engineering 139 (1-4) (1996) 195–227.

[101] G. Abaqus, Abaqus 6.11, Dassault Systemes Simulia Corporation, Providence, RI, USA.

[102] R. B. Lehoucq, P. L. Reu, D. Z. Turner, A novel class of strain measures for digital image correlation, Strain 51 (4) (2015) 265–275.

[103] G. Kissas, J. Seidman, L. F. Guilhoto, V. M. Preciado, G. J. Pappas, P. Perdikaris, Learning operators with coupled attention, arXiv preprint arXiv:2201.01032.