HHS Public Access

Author manuscript

Comput Methods Appl Mech Eng. Author manuscript; available in PMC 2024 December 15.

Published in final edited form as:

Comput Methods Appl Mech Eng. 2023 December 15; 417(Pt B): . doi:10.1016/j.cma.2023.116280.

MetaNO: How to Transfer Your Knowledge on Learning Hidden Physics

Lu Zhang[a], **Huaiqian You**[a], **Tian Gao**[b], **Mo Yu**[c], **Chung-Hao Lee**[d], **Yue Yu**[a,**]

[a]Department of Mathematics, Lehigh University, Bethlehem, PA, USA

[b]IBM Research, Yorktown Heights, NY, USA

[c]Pattern Recognition Center, WeChat AI, Tencent Inc, China

[d]School of Aerospace and Mechanical Engineering, The University of Oklahoma, Norman, OK, USA

Abstract

Gradient-based meta-learning methods have primarily been applied to classical machine learning tasks such as image classification. Recently, PDE-solving deep learning methods, such as neural operators, are starting to make an important impact on learning and predicting the response of a complex physical system directly from observational data. Taking the material modeling problems for example, the neural operator approach learns a surrogate mapping from the loading field to the corresponding material response field, which can be seen as learning the solution operator of a hidden PDE. The microstructure and mechanical parameters of each material specimen correspond to the (possibly heterogeneous) parameter field in this hidden PDE. Due to the limitation on experimental measurement techniques, the data acquisition for each material specimen is commonly challenging and costly. This fact calls for the utilization and transfer of existing knowledge to new and unseen material specimens, which corresponds to sampling efficient learning of the solution operator of a hidden PDE with a different parameter field. Herein, we propose a novel meta-learning approach for neural operators, which can be seen as transferring the knowledge of solution operators between governing (unknown) PDEs with varying parameter fields. Our approach is a provably universal solution operator for multiple PDE solving tasks, with a key theoretical observation that underlying parameter fields can be captured in the first layer of neural operator models, in contrast to typical final-layer transfer in existing meta-learning methods. As applications, we demonstrate the efficacy of our proposed approach on PDE-based datasets and a real-world material modeling problem, illustrating that our method can handle

**Corresponding author: yuy214@lehigh.edu (Yue Yu).

*We dedicate this paper to Prof. Thomas J.R. Hughes for his lifetime achievements, pioneering contributions to computational mechanics and applied mathematics, and distinguished editorship of Computer Methods in Applied Mechanics and Engineering for more than four decades.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

complex and nonlinear physical response learning tasks while greatly improving the sampling efficiency in unseen tasks.

Keywords

Operator-Regression Neural Networks; Data-Driven Physics Modeling; Neural Operators; Transfer Learning; Meta-Learning; Scientific Machine Learning

1. Introduction

Few-shot learning is an important problem in machine learning, where new tasks are learned with a very limited number of labelled datapoints [1]. In recent years, significant progress has been made on few-shot learning using meta-learning approaches [2–12]. Broadly speaking, given a family of tasks, some of which are used for training and others for testing, meta-learning approaches aim to learn a shared multi-task representation that can generalize across the different training tasks, and result in fast adaptation to new and unseen testing tasks. Meta-learning learning algorithms have been successfully applied to conventional machine learning problems such as image classification, function regression, and reinforcement learning, but studies on few-shot learning approaches for complex physical system modeling problems have been limited. The call of developing a few-shot learning approach for complex physical system modeling problems, such as to learn the material response model from experimental measurements, is just as acute, while the typical understanding of how multi-task learning should be applied on this scenario is still nascent.

As a motivating example, we consider the scenario of new material discovery in the lab environment, where the material model is built based on experimental measurements of its responses subject to different loadings. Since the physical properties (such as the mechanical and structural parameters) in different material specimens vary, the model learnt from experimental measurements on one specimen would have large generalization errors on other specimens. As a result, the data-driven model has to be trained repeatedly with a large number of material specimens, which makes the learning process inefficient. Furthermore, experimental measurement acquisition of these specimens is often challenging and expensive. In some problems, a large amount of measurements are not even feasible. For example, in the design and testing of biosynthetic tissues, performing repeated loading would potentially induce the cross-linking and permanent set phenomenon, which notoriously alter the tissue durability [13]. As a result, it is critical to learn the physical response model of a new specimen with sample size as small as possible. Furthermore, since many characterization methods to obtain underlying material mechanistic and structural properties would require the use of destructive methods [14, 15], in practice many physical properties are not measured and can only be treated as hidden and unknown variables. Hence, we likely only have limited access to the measurements on the complex system responses caused by the change of these physical properties.

Supervised operator learning methods are typically used to address this class of problems. They take a number of observations on the loading field as input, and try to predict the corresponding physical system response field as output, corresponding to one underlying

PDE (as one task). Herein, we consider the meta-learning of multiple complex physical systems (as tasks), such that all these tasks are governed by a common PDE with different (hidden) physical property or parameter fields. Formally, assume that we have a distribution $p(\mathcal{T})$ over tasks, each task $\mathcal{T}^\eta \sim p(\mathcal{T})$ corresponds to a hidden physical property field $\boldsymbol{b}^\eta(\boldsymbol{x}) \in \mathcal{B}\left(\mathbb{R}^{d_b}\right)$ that contains the task-specific mechanistic and structural information in our material modeling example. On task $\mathcal{T}^\eta$, we have a number of observations on the loading field $\boldsymbol{g}_i^\eta(\boldsymbol{x}) \in \mathcal{A}\left(\mathbb{R}^{d_g}\right)$ and the corresponding physical system response field $\boldsymbol{u}_i^\eta(\boldsymbol{x}) \in \mathcal{U}\left(\mathbb{R}^{d_u}\right)$ according to a hidden parameter field $\boldsymbol{b}^\eta(\boldsymbol{x})$. Here, i is the sample index, $\mathcal{B}, \mathcal{A}$ and $\mathcal{U}$ are Banach spaces of function taking values in $\mathbb{R}^{d_b}, \mathbb{R}^{d_g}$ and $\mathbb{R}^{d_u}$, respectively. For task $\mathcal{T}^\eta$, our modeling goal is to learn the solution operator $\mathcal{G}_{b^\eta}^\eta: \mathcal{A} \to \mathcal{U}$, such that the learnt model can predict the corresponding physical response field $\boldsymbol{u}(\boldsymbol{x})$ for any loading field $\boldsymbol{g}(\boldsymbol{x})$[1]. Without transfer learning, one needs to learn a surrogate solution operator for each task only based on the data pairs on this task, and repeat the training for every task. The learning procedure would require a relatively large amount of observation pairs and training time for each task. Therefore, this physical-based modeling scenario raises a key question: *Given data from a number of parametric PDE solving (training) tasks with different unknown parameters, how can one efficiently learn an accurate surrogate solution operator for a test task with new and unknown parameters, with few data on this task*[2] *?*

To address this question, we introduce MetaNO, a novel meta-learning approach for transferring knowledge between neural operators, which can be seen as transferring the knowledge of solution operators between governing (potentially unknown) PDEs with varying hidden parameter fields. Our **main contributions** are:

- MetaNO is the first neural-operator-based meta-learning approach for multiple tasks, which not only preserves the generalizability to different resolutions and input functions from the integral neural operator architecture, but also improves sampling efficiency on new tasks – for comparable accuracy, MetaNO saves the number of measurements required by ~90%.
- With rigorous operator approximation analysis, we made the key observation that the hidden parameter field can be captured by adapting the first layer of the neural operator model. Therefore, our MetaNO is *substantially different* from existed popular meta-learning approaches [5, 10], since the later typically rely on the adaptation of their last layers [12]. By construction, MetaNO serves as a provably universal solution operator for multiple PDE solving tasks.
- On synthetic, benchmark, and real-world biological tissue datasets, the proposed method consistently outperforms existing non-meta transfer-learning baselines and other gradient-based meta-learning methods.

[1]Without ambiguity, in the following context we will neglect the subscript and denote $\mathcal{G}_{b^\eta}^\eta$ as $\mathcal{G}^\eta$ for notation simplicity.

[2]In some meta-learning literature, e.g., [16], these small sets of labelled data pairs on a new task (or any task) is called the context, and the learnt model will be evaluated on an additional set of unlabelled data pairs, i.e., the target.

2. Background and Related Work

2.1. Hidden Physics Learning with Neural Networks

For many decades, physics-based PDEs have been commonly employed for predicting and monitoring complex system responses. Then traditional numerical methods were developed to solve these PDEs and provide predictions for desired system responses. However, three fundamental challenges usually present. First, the choice of governing PDE laws is often determined *a priori* and free parameters are often tuned to obtain agreement with experimental data, which makes the rigorous calibration and validation process challenging. Second, traditional numerical methods are solved for specific boundary and initial conditions, as well as loading or source terms. Therefore, they are not generalizable for other operating conditions and hence not effective for real-time prediction. Third, complex PDE systems such as turbulence flows and heterogeneous materials modeling problems usually require a very fine discretization, and are therefore very time-consuming for traditional solvers.

To provide an efficient surrogate model for physical responses, machine learning methods may hold the key. Recently, there has been significant progress in the development of deep neural networks (NNs) for learning the hidden physics of a complex system [17–25]. Among these methods, the neural operators show particular promises in resolving the above challenges, which aim to learn mappings between inputs of a dynamical system and its state, so that the network can serve as a surrogate for a solution operator [26–34].

Comparing with classical NNs, most notable advantages of neural operators are resolution independence and generalizability to different input instances. Moreover, comparing with the classical PDE modeling approaches, neural operators require only data with no knowledge of the underlying PDE. All these advantages make neural operators promising tools to PDE learning tasks. Examples include modeling the unknown physics law of real-world problems [35, 36] and providing efficient solution operator for PDEs [26–28, 37, 38]. On the other hand, data in scientific applications are often scarce and incomplete. Utilization of other relevant data sources could alleviate such a problem, yet no existing work have addressed the transferability of neural operators. Through the meta-learning techniques, our work fulfills the demand of such a transfer setting, with the same type of PDE system but different (hidden) physical properties.

2.2. Base Model: Integral Neural Operators

We briefly introduce the integral neural operator model, which will be utilized as the base model of this work. The integral neural operators, first proposed in [26] and further developed in [27–29, 39] comprises of three building blocks. First, the input function, $\boldsymbol{g}(\boldsymbol{x}) \in \mathscr{A}$, is lifted to a higher dimensional representation via $\boldsymbol{h}(\boldsymbol{x}, 0) = \mathscr{P}[\boldsymbol{g}](\boldsymbol{x}) := \boldsymbol{P}(\boldsymbol{x})[\boldsymbol{x}, \boldsymbol{g}(\boldsymbol{x})]^T + \boldsymbol{p}(\boldsymbol{x})$. $\boldsymbol{P}(\boldsymbol{x}) \in \mathbb{R}^{(s+d_g)\times d_h}$ and $\boldsymbol{p}(\boldsymbol{x}) \in \mathbb{R}^{d_h}$ define an affine pointwise mapping, which are often taken as constant parameters, i.e., $\boldsymbol{P}(\boldsymbol{x}) \equiv \boldsymbol{P}$ and $\boldsymbol{p}(\boldsymbol{x}) \equiv \boldsymbol{p}$. Then, the feature vector function $\boldsymbol{h}(\boldsymbol{x}, 0)$ goes through an iterative layer block where the layer update is defined via the action of the sum of a local linear operator, a nonlocal integral kernel operator, and a bias function: $\boldsymbol{h}(\cdot, l+1) = \mathscr{J}_{l+1}[\boldsymbol{h}(\cdot, l)]$. Here,

$\boldsymbol{h}(\cdot, l), l \in \{0, \cdots, L\}$, is a sequence of functions representing values of the network at each hidden layer, taking values in $\mathbb{R}^{d_h}$. $\mathcal{J}_1, \cdots, \mathcal{J}_L$ are nonlinear operator layers. In this work, we employ the implicit Fourier neural operator (IFNO) as the base model[3] and take the iterative layers as $\mathcal{J}_1 = \cdots = \mathcal{J}_L = \mathcal{J}$, where

$$\boldsymbol{h}(\boldsymbol{x}, l+1) = \mathcal{J}[\boldsymbol{h}(\boldsymbol{x}, l)] := \boldsymbol{h}(\boldsymbol{x}, l) + \frac{1}{L}\sigma\Big(W\boldsymbol{h}(\boldsymbol{x}, l) + \mathcal{F}^{-1}[\mathcal{F}[\kappa(\cdot\,; \boldsymbol{v})] \cdot \mathcal{F}[\boldsymbol{h}(\cdot, l)]](\boldsymbol{x}) + \boldsymbol{c}(\boldsymbol{x})\Big). \tag{2.1}$$

$\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier transform and its inverse, respectively. $\boldsymbol{c} \in \mathbb{R}^{d_h}$ defines a constant bias, $W \in \mathbb{R}^{d_h \times d_h}$ is the weight matrix, and $\mathcal{F}[\kappa(\cdot\,; \boldsymbol{v})] := R$ is a circulant matrix that depends on the convolution kernel κ. σ is an activation function, which is often taken to be the popular rectified linear unit (ReLU) function. Finally, the output $\boldsymbol{u}(\cdot) \in \mathcal{U}$ is obtained through a projection layer, by mapping the last hidden layer representation $\boldsymbol{h}(\cdot, L)$ onto $\mathcal{U}$ as:
$\boldsymbol{u}(\boldsymbol{x}) = \mathcal{Q}[\boldsymbol{h}(\cdot, L)](\boldsymbol{x}) := Q_2(\boldsymbol{x})\sigma(Q_1\boldsymbol{h}(\boldsymbol{x}, L) + \boldsymbol{q}_1(\boldsymbol{x})) + \boldsymbol{q}_2(\boldsymbol{x})$. $Q_1(\boldsymbol{x}) \in \mathbb{R}^{d_Q \times d_h}, Q_2(\boldsymbol{x}) \in \mathbb{R}^{d_u \times d_Q}, \boldsymbol{q}_1(\boldsymbol{x}) \in \mathbb{R}^{d_Q}$
and $\boldsymbol{q}_2(\boldsymbol{x}) \in \mathbb{R}^{d_u}$ are appropriately sized matrices and vectors that are part of the parameter set that we aim to learn, which are often taken as constant parameters and will be denoted as $Q_1, Q_2, \boldsymbol{q}_1$ and $\boldsymbol{q}_2$, respectively. In the following, we denote the set of trainable parameters in the lifting layer as θ_P, the set from the iterative layer block as θ_I, and the set in the projection layer as θ_Q.

The neural operator can be employed to learn an approximation for the solution operator, $\mathcal{G}$. Given $\mathcal{D} := \{(\boldsymbol{g}_i, \boldsymbol{u}_i)\}_{i=1}^N$, a labelled (context) set of observations, where the input $\{\boldsymbol{g}_i\} \subset \mathcal{A}$ is a set of independent and identically distributed (i.i.d.) random fields from a known probability distribution μ on $\mathcal{A}$, and $\boldsymbol{u}_i(\boldsymbol{x}) \in \mathcal{U}$ is the observed but possibly noisy corresponding solution. Let $\Omega \subset \mathbb{R}^s$ be the domain of interest, we assume that all observations can be modeled with a parametric PDE form:

$$\mathcal{K}_{\boldsymbol{b}(\boldsymbol{x})}[\boldsymbol{u}_i](\boldsymbol{x}) = \boldsymbol{g}_i(\boldsymbol{x}), \ \boldsymbol{x} \in \Omega. \tag{2.2}$$

$\mathcal{K}_{\boldsymbol{b}}$ is the operator representing the possibly unknown governing law, e.g., balance laws. Then, the system response can be learnt by constructing a surrogate solution operator of (2.2): $\widetilde{\mathcal{G}}[\boldsymbol{g}; \theta](\boldsymbol{x}) := \mathcal{Q}_{\theta_Q} \circ (\mathcal{J}_{\theta_I})^L \circ \mathcal{P}_{\theta_P}[\boldsymbol{g}](\boldsymbol{x}) \approx \boldsymbol{u}(\boldsymbol{x})$, where parameter set $\theta = [\theta_P, \theta_I, \theta_Q]$ is obtained by solving the optimization problem:

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

[3] We also point out that the proposed multi-task strategy is generic and hence also applicable to other neural operators [26–29, 32].

(2.3)

Here C denotes a properly defined cost functional which is often taken as the relative mean square error.

2.3. Gradient-Based Meta-Learning Methods

Traditionally, machine learning models are designed to address one single task by learning from the dataset corresponding to such a task. Hence, its efficacy relies on the quality and abundance of available data. To improve the learning efficacy in the small data regime, a significant body of work has tackled the challenge of few-shot learning using transfer-learning or meta-learning approaches, with the basic idea of utilizing prior knowledge gained from related (source) tasks to improve the learning performance on the target task. Among these two types of approaches, the basic idea of transfer-learning is to train a model from the source task and then refine the pre-trained model in the target task [40]. In comparison, meta-learning approaches aim to learn a generalizable representations that shared across the different training tasks, which can accelerate learning of test tasks [5, 41]. One of highly successful meta-learning algorithms is Model Agnostic Meta-Learning (MAML) [5], which led to the development of a series of related gradient-based meta-learning (GBML) methods [7, 9, 10, 42]. Almost-No-Inner-Loop algorithm (ANIL) [10] modifies MAML by freezing the final layer representation during local adaptation. Recently, theoretical analysis [12] found that the driving force causing MAML and ANIL to recover the general representation is the adaptation of the final layer of their models, which harnesses the underlying task diversity to improve the representation in all directions of interest.

Beyond applications such as image classification and reinforcement learning, a few approaches have studied hidden-physics learning under meta [43–46] or even transfer setting [47, 48]. Among these meta-learning works, [43, 44] are designed for specific physical applications, while [45, 46] focus on on dynamics forecasting by learning the temporal evolution information directly [45] or learning time-invariant features [46]. Hence, none of these works have provided a generic approach nor theoretical understanding on how to transfer the multi-task knowledge between a series of complex physical systems, such that all these tasks are governed by a common parametric PDE with different physical parameters.

3. Meta-Learnt Neural Operator

To transfer the multi-task knowledge between a series of complex systems governed by different hidden physical parameters, we proposed to leverage the integral neural operator with a meta-learning setting. Before elaborating our novel meta-learnt neural operator architecture, MetaNO, we formally state the transfer-learning problem setting for PDE with different parameters.

Assume that we have a set of training tasks $\{\mathcal{T}^\eta\}$ such that $\mathcal{T}^\eta \sim p(\mathcal{T})$, and for each training task we have a set of observations of loading field/respond field data pairs $\mathcal{D}^\eta := \{(\boldsymbol{g}_i^\eta(\boldsymbol{x}), \boldsymbol{u}_i^\eta(\boldsymbol{x}))\}_{i=1}^{N^\eta}$. Each task can be modeled with a parametric PDE form

$$\mathcal{K}_{\boldsymbol{b}^\eta(\boldsymbol{x})}[\boldsymbol{u}_i^\eta](\boldsymbol{x}) = \boldsymbol{g}_i^\eta(\boldsymbol{x}), \ \boldsymbol{x} \in \Omega, \tag{3.1}$$

where $\boldsymbol{b}^\eta(\boldsymbol{x})$ is the hidden task-specific physical parameter field for the common governing law. Given a new and unseen test task, $\mathcal{T}^{\text{test}}$, and a (usually small) context set of labelled samples $\mathcal{D}^{\text{test}} := \{(\boldsymbol{g}_i^{\text{test}}(\boldsymbol{x}), \boldsymbol{u}_i^{\text{test}}(\boldsymbol{x}))\}_{i=1}^{N^{\text{test}}}$ on it, our goal is to obtain the approximated solution operator model on the test task as $\widetilde{\mathcal{G}}\left[\boldsymbol{g}; \theta^{\text{test}}\right]$. To provide a quantitative metric of the performance for each method, we reserve a separate set of labelled samples on the test task as the target set, and measure averaged relative errors of $\boldsymbol{u}$ on this set. In the few-shot learning context, we are particularly interested in the small-sample scenario where $N^{\text{test}} \ll N^\eta$.

3.1. A Novel Meta-Learnt Neural Operator Architecture

We now propose MetaNO, which applies *task-wise adaptation only to the first layer*, *i.e., the lifting layer*, with the full algorithm outlined in Algorithm 1. We point out that MetaNO is substantially different from existed popular meta-learning approaches such as MAML and ANIL, since the later rely on the adaptation of their last layer, as shown in [12]. This property makes MetaNO more suitable for PDE solving tasks as will be discussed in theoretical analysis below and confirmed in empirical evaluations of Section 4.

Algorithm 1

MetaNO

Meta-Train Phase:

Input: a batch $\left\{\mathcal{T}^\eta\right\}_{\eta=1}^{H}$ of training tasks and labelled data pairs $\mathcal{D}^\eta := \{(\boldsymbol{g}_i^\eta(\boldsymbol{x}), \boldsymbol{u}_i^\eta(\boldsymbol{x}))\}_{i=1}^{N^\eta}$ on each task.

Output: common parameters θ_I^* and θ_Q^* across all tasks.

1. Initialize θ_I, θ_Q, and $\{\theta_P^\eta\}_{\eta=1}^{H}$.

2. Solve for $\left[\left\{\theta_P^{\eta,*}\right\}_{\eta=1}^{H}, \theta_I^*, \theta_Q^*\right]$ from the optimization problem in (3.2).

Meta-Test Phase:

Input: a test task $\mathcal{T}^{\text{test}}$ and few labelled data pairs $\mathcal{D}^{\text{test}} := \{(\boldsymbol{g}_i^{\text{test}}(\boldsymbol{x}), \boldsymbol{u}_i^{\text{test}}(\boldsymbol{x}))\}_{i=1}^{N^{\text{test}}}$ on it.

Output: the task-wise parameter $\theta_P^{\text{test},*}$ and the corresponding surrogate PDE solution operator $\widetilde{\mathcal{G}}\left[\boldsymbol{g}; \left[\theta_P^{\text{test},*}, \theta_I^*, \theta_Q^*\right]\right](\boldsymbol{x})$ for the test task.

3. Solve for the lift layer parameter $\theta_P^{\text{test},*}$ from the optimization problem in (3.3).

4. *(For cases with large* N^{test} *and/or small* N^η*)*, fine tune all parameters on the test task.

Similar as in other meta-learning approaches [49–52], the MetaNO algorithm consists of two phases: 1) a meta-train phase which learns shared iterative layers parameters θ_I and projection layer parameters θ_P from training tasks; 2) a meta-test phase which transfers the learned knowledge and rapidly learning surrogate solution operators for unseen test tasks with unknown physical parameter field, where only a few labelled samples are provided. In the meta-train phase, a batch $\{\mathscr{T}^\eta\}_{\eta=1}^H$ of H tasks is drawn from the training tasks set, with a context set of N^η numbers of labelled loading field/response field data pairs, $\mathscr{D}^\eta := \{(\boldsymbol{g}_i^\eta(\boldsymbol{x}), \boldsymbol{u}_i^\eta(\boldsymbol{x}))\}_{i=1}^{N^\eta}$, provided on each task. Then, we seek the common iterative (θ_I) and projection (θ_Q) parameters, and the task-wise lifting parameters θ_P^η by solving the optimization problem:

$$\left[\{\theta_P^{\eta,*}\}_{\eta=1}^H, \theta_I^*, \theta_Q^*\right] = \underset{\left\{\{\theta_P^\eta\}_{\eta=1}^H, \theta_I, \theta_Q\right\}}{\operatorname{argmin}} \sum_{\eta=1}^H \mathscr{L}_{\mathscr{D}^\eta}([\theta_P^\eta, \theta_I, \theta_Q]). \tag{3.2}$$

Then, in the meta-test phase, we adapt the knowledge to a new and unseen test task $\mathscr{T}^{\text{test}}$, with limited data on the context set $\mathscr{D}^{\text{test}} := \{(\boldsymbol{g}_i^{\text{test}}(\boldsymbol{x}), \boldsymbol{u}_i^{\text{test}}(\boldsymbol{x}))\}_{i=1}^{N^{\text{test}}}$ on this task. In particular, we fix the common parameters θ_I^* and θ_Q^*, then solve for the task-wise parameter θ_P^{test} via:

$$\theta_P^{\text{test},*} = \underset{\theta_P^{\text{test}}}{\operatorname{argmin}}\, \mathscr{L}_{\mathscr{D}^{\text{test}}}\left(\left[\theta_P^{\text{test}}, \theta_I^*, \theta_Q^*\right]\right). \tag{3.3}$$

One can then fine tune all test task parameters $[\theta_P^{\text{test}}, \theta_I, \theta_Q]$ for further improvements. Finally, the surrogate PDE solution operator on the test task is obtained as:

$$\widetilde{\mathscr{G}}\left[\boldsymbol{g}; \left[\theta_P^{\text{test},*}, \theta_I^*, \theta_Q^*\right]\right](\boldsymbol{x}) := \mathscr{Q}_{\theta_Q^*} \circ (\mathscr{J}_{\theta_I^*})^L \circ \mathscr{P}_{\theta_P^{\text{test},*}}[\boldsymbol{g}](\boldsymbol{x}).$$

and will be evaluated on a reserved target data set on the test task.

3.2. Universal Solution Operator

To see the inspiration of the proposed architecture, without loss of generality, we assume that the underlying task parameter field $\boldsymbol{b}^\eta(\boldsymbol{x})$, modeling the physical property field, is normalized and satisfying $\|\boldsymbol{b}^\eta(\boldsymbol{x}) - \bar{\boldsymbol{b}}(\boldsymbol{x})\|_{L^2(\Omega)} \le 1$ for all $\eta \in \{1, \cdots, H\}$, where $\bar{\boldsymbol{b}} := \mathbb{E}_{\mathscr{T}^\eta \sim p(\mathscr{T})}[\boldsymbol{b}^\eta]$. Denoting $\mathscr{F}_{\boldsymbol{u}}[\boldsymbol{b}] := \mathscr{K}_{\boldsymbol{b}}[\boldsymbol{u}]$ as a function from physical parameter fields $\mathscr{B}$ to loading fields $\mathscr{A}$, we take the Fréchet derivative of $\mathscr{F}$ with respect to $\boldsymbol{b} - \bar{\boldsymbol{b}}$ and obtain:

$$\mathscr{K}_{\boldsymbol{b}^\eta}[\boldsymbol{u}] = \mathscr{F}_{\boldsymbol{u}}[\bar{\boldsymbol{b}}] + D\mathscr{F}_{\boldsymbol{u}}[\bar{\boldsymbol{b}}](\boldsymbol{b}^\eta - \bar{\boldsymbol{b}}) + o(\|\boldsymbol{b}^\eta - \bar{\boldsymbol{b}}\|_{L^2(\Omega)}).$$

Substituting the above formulation into (3.1) yields:

$$\mathscr{F}_{u_i^\eta}[\bar{b}] + D\mathscr{F}_{u_i^\eta}[\bar{b}](b^\eta - \bar{b}) \approx g_i^\eta .$$

Denoting $F_1[b^\eta] := [1, b^\eta - \bar{b}]$ and $F_2[u_i^\eta] := [\mathscr{F}_{u_i^\eta}[\bar{b}], D\mathscr{F}_{u_i^\eta}[\bar{b}]]$, we can reformulate (3.1) into a more generic form:

$$F_1[b^\eta](x) \cdot F_2[u_i^\eta](x) = g_i^\eta(x), \ x \in \Omega .$$

Note that this parametric PDE form is very general and applicable to many science and engineering applications – besides our motivating example on material modeling, other examples include the monitoring of tissue degeneration problems [13], the detection of subsurface flows [53], the nondestructive inspection in aviation [54], and the prediction of concrete structures deterioration [55], etc.

In the following, we show that MetaNOs are universal solution operators for the multi-task PDE solving problem in (3.4), in the sense that they can approximate a fixed point method to a desired accuracy. For simplicity, we consider a $1D$ domain $\Omega \subset \mathbb{R}$, and scalar-valued functions $F_1[b^\eta]$, $F_2[u_i^\eta]$. These functions are assumed to be sufficiently smooth and measured at uniformly distributed nodes $\chi := \{x_1, x_2, \ldots, x_M\}$, with $F_1[b^\eta](x_j) \neq 0$ for all η and j. Then, (3.4) can be formulated as an implicit system of equations:

$$\mathscr{H}(U_i^{\eta,*}; \widetilde{G}_i^\eta) := \begin{bmatrix} F_2[u_i^\eta](x_1) - g_i^\eta(x_1)/F_1[b^\eta](x_1) \\ \vdots \\ F_2[u_i^\eta](x_M) - g_i^\eta(x_M)/F_1[b^\eta](x_M) \end{bmatrix} = \mathbf{0}, \tag{3.5}$$

where $U_i^{\eta,*} := [u_i^\eta(x_1), \ldots, u_i^\eta(x_M)]$ is the solution we seek, $\widetilde{G}_i^\eta := [g_i^\eta(x_1)/F_1[b^\eta](x_1), \ldots, g_i^\eta(x_M)/F_1[b^\eta](x_M)]$ is the reparameterized loading vector, and $G_i^\eta := [g_i^\eta(x_1), g_i^\eta(x_2), \ldots, g_i^\eta(x_M)]$ is the original loading vector. Here, we notice that all task-specific information is encoded in $\widetilde{G}_i^\eta$ and can be captured in the lifting layer parameter. Therefore, when seeing (3.5) as an implicit problem of $U_i^{\eta,*}$ and $\widetilde{G}_i^\eta$, it is actually independent of the task parameter field b^η, i.e., this problem is task-independent. In the following, we refer to (3.5) without the task index, as $\mathscr{H}(U^*; \widetilde{G})$, for notation simplicity.

To solve for U^* from the nonlinear system $\mathscr{H}(U^*; \widetilde{G}) = 0$, a popular approach would be to use fixed-point iteration methods such as the Newton-Raphson method. With an initial guess of the solution (denoted as U^0), the process is repeated to produce successively better approximations to the roots of (3.5), from the solution of iteration l (denoted as U^l) to that of $l + 1$ (denoted as U^{l+1}) as:

$$\boldsymbol{U}^{l+1} = \boldsymbol{U}^{l} - \left(\nabla \mathscr{H}\left(\boldsymbol{U}^{l}; \widetilde{\boldsymbol{G}}\right)\right)^{-1} \mathscr{H}\left(\boldsymbol{U}^{l}; \widetilde{\boldsymbol{G}}\right) := \boldsymbol{U}^{l} + \mathscr{R}\left(\boldsymbol{U}^{l}, \widetilde{\boldsymbol{G}}\right), \tag{3.6}$$

until a sufficiently precise value is reached. In the following, we show that as long as Assumptions 1 and 2 hold, i.e., there exists a converging fixed point method, then MetaNO can be seen as an resemblance of the fixed point method in (3.6) and hence acts as an universal approximator of the solution operator for (3.4).

Assumption 1. There exists a fixed point equation, $\boldsymbol{U} = \boldsymbol{U} + \mathscr{R}\left(\boldsymbol{U}, \widetilde{\boldsymbol{G}}\right)$ for the implicit of problem (3.5), such that $\mathscr{R}: \mathbb{R}^{2M} \mapsto \mathbb{R}^{M}$ is a continuous function satisfying $\mathscr{R}\left(\boldsymbol{U}, \widetilde{\boldsymbol{G}}\right) = 0$ and $\| \mathscr{R}\left(\widehat{\boldsymbol{U}}, \widetilde{\boldsymbol{G}}\right) - \mathscr{R}\left(\widetilde{\boldsymbol{U}}, \widetilde{\boldsymbol{G}}\right) \|_{l^2\left(\mathbb{R}^M\right)} \leq m \| \widehat{\boldsymbol{U}} - \widetilde{\boldsymbol{U}} \|_{l^2\left(\mathbb{R}^M\right)}$ for any two vectors $\widehat{\boldsymbol{U}}, \widetilde{\boldsymbol{U}} \in \mathbb{R}^{M}$. Here, $m \geq 0$ is a constant independent of $\widetilde{\boldsymbol{G}}$.

Assumption 2. With the initial guess $\boldsymbol{U}^{0} := [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_M]$, the fixed-point iteration $\boldsymbol{U}^{l+1} = \boldsymbol{U}^{l} + \mathscr{R}\left(\boldsymbol{U}^{l}, \widetilde{\boldsymbol{G}}\right)$ $(l = 0, 1, \ldots)$ converges, i.e., for any given $\varepsilon > 0$, there exists an integer L such that

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

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

Intuitively, Assumptions 1 and 2 ensure the hidden PDEs to be numerically solvable with a converging iterative solver, which is a typical required condition of numerical PDE solving problems. Then, we have our universal approximation theorem as below. The main result of this theorem is to show that for any desired accuracy $\varepsilon > 0$, one can find a sufficiently large $L > 0$ and sets of parameters $\theta^{\eta} = \{\theta_P^{\eta}, \theta_I, \theta_Q\}$, such that the resultant MetaNO model acts as a fixed point method with the desired prediction for all tasks and samples.

Theorem 1 (Universal approximation). Given Assumptions 1–2, let 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 $d_h > 0$, such that one can find a parameter set for the multi-task problem, $\theta^{\eta} = [\theta_P^{\eta}, \theta_I, \theta_Q]$, such that the corresponding MetaNO model satisfies

$$\left\| \mathscr{Q}_{\theta_Q} \circ \left(\mathscr{J}_{\theta_I}\right)^{L} \circ \mathscr{P}_{\theta_P^{\eta}}\left(\left[\boldsymbol{U}^{0}, \boldsymbol{G}^{\eta}\right]^{\mathrm{T}}\right) - \boldsymbol{U}^{\eta, *} \right\| \leq \varepsilon,$$

for all loading instance $\boldsymbol{G}^{\eta} \in \mathbb{R}^{M}$ and tasks.

We now provide the detailed proof for Theorem 1, based on Assumptions 1 and 2. Intuitively, these assumptions mean the underlying implicit problem is solvable with a

converging fixed point method. This condition is a basic requirement by numerical PDEs, and it generally holds true in many applications governed by nonlinear and complex PDEs, such as in our three experiments.

Here, we prove that the MetaNO is universal, i.e., given a fixed point method satisfying Assumptions 1 and 2, one can find parameter sets θ^{η} whose output approximates $\boldsymbol{U}^{\eta,*}$ to a desired accuracy, $\varepsilon > 0$, for all $\eta = 1, \cdots, H$ tasks. For the task-wise parameters, with a slight abuse of notation, we denote $P^{\eta} \in \mathbb{R}^{d_h M \times (d_g + s)M}$ as the collection of the pointwise weight matrices at each discretization point in χ for the η-th task, and $\boldsymbol{p}^{\eta} \in \mathbb{R}^{d_h M}$ for the bias in the lifting layer. Then, for the parameters shared among all tasks, in the iterative layer we denote $\boldsymbol{C} = [\boldsymbol{c}(\boldsymbol{x}_1), \cdots, \boldsymbol{c}(\boldsymbol{x}_M)] \in \mathbb{R}^{d_h M}$ as the collection of pointwise bias vectors $\boldsymbol{c}(\boldsymbol{x}_i), W \in \mathbb{R}^{d_h \times d_h}$ for the local linear transformation, and $R = \mathscr{F}\left[\kappa\left(\cdot\,; \boldsymbol{v}\right)\right] \in \mathbb{C}^{d_h \times d_h \times M} \in \mathbb{C}^{d_h \times d_h \times M}$ for the Fourier coefficients of the kernel κ. For simplicity, here we have assumed that the Fourier coefficient is not truncated, and all available frequencies are used. Then, for the projection layer we seek $Q_1 \in \mathbb{R}^{d_Q M \times d_h M}$, $Q_2 \in \mathbb{R}^{d_u M \times d_Q M}$, $\boldsymbol{q}_1 \in \mathbb{R}^{d_Q M}$ and $\boldsymbol{q}_2 \in \mathbb{R}^{d_u M}$. For the simplicity of notation, in this section we organize the feature vector $\boldsymbol{H} \in \mathbb{R}^{d_h M}$ 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 $\boldsymbol{H}(\boldsymbol{x}_i) \in \mathbb{R}^{d_h}$.

We point out that under this circumstance, the (discretized) iterative layer can be written as

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

with

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

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

$$\mathscr{J}(\boldsymbol{H}(l)) := \boldsymbol{H}(l) + \frac{1}{L}\sigma\left(\widetilde{V}\boldsymbol{H}(l) + \boldsymbol{C}\right)$$

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

For the proof of this main theorem, we need the following approximation property of a shallow neural network, with its detailed proof provided in [39]:

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

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

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 $\boldsymbol{X}$.

We now proceed to the proof of Theorem 1:

Proof. Since all $\boldsymbol{U}^{\eta,*}$ satisfies Assumptions 1–2, for any $\varepsilon > 0$, we first pick a sufficiently large integer L such that the L-th layer iteration result of this fixed point formulation satisfies $\left\| \boldsymbol{U}^L - \boldsymbol{U}^{\eta,*} \right\|_{l^2\left(\mathbb{R}^M\right)} \leq \frac{\varepsilon}{2}$ for all tasks. 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 $\hat{\mathscr{R}}\left(\boldsymbol{U}^\eta, \widetilde{\boldsymbol{G}}^\eta\right) := S\sigma\left(B\left[\boldsymbol{U}^\eta, \widetilde{\boldsymbol{G}}^\eta\right]^{\mathrm{T}} + A\right)$ satisfies

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

where m is the contraction parameter of $\mathscr{R}$, 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

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

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 parameters of MetaNO as follows. In this construction, we choose the feature dimension as $d_h := \tilde{d}M$. With the input $[\boldsymbol{U}^0, \boldsymbol{G}^\eta] \in \mathbb{R}^{2M}$, for the lift layer we set

$$P^\eta := 1_{[M,1]} \otimes \begin{bmatrix} S^+ & \mathbf{0} \\ \mathbf{0} & D^\eta \end{bmatrix} = \underbrace{\begin{bmatrix} S^+ & \mathbf{0} & S^+ & \mathbf{0} & \cdots & S^+ & \mathbf{0} \\ \mathbf{0} & D^\eta & \mathbf{0} & D^\eta & \cdots & \mathbf{0} & D^\eta \end{bmatrix}^T}_{\text{repeated for } M \text{ times}} \in \mathbb{R}^{d_h M \times 2M},$$

and $\boldsymbol{p}^\eta := \mathbf{0} \in \mathbb{R}^{d_h M}$. Here, $D^\eta := \mathrm{diag}[1/\boldsymbol{F}_1[\boldsymbol{b}^\eta](\boldsymbol{x}_1), \cdots, 1/\boldsymbol{F}_1[\boldsymbol{b}^\eta](\boldsymbol{x}_M)]$. As such, the initial layer of feature is then given by

$$\boldsymbol{H}(0) = P^\eta\left([\boldsymbol{U}^0, \boldsymbol{G}^\eta]^{\mathrm{T}}\right) = 1_{[M,1]} \otimes [S^+\boldsymbol{U}^0, D^\eta \boldsymbol{G}^\eta]^{\mathrm{T}} = 1_{[M,1]} \otimes [S^+\boldsymbol{U}^0, \widetilde{\boldsymbol{G}}^\eta]^{\mathrm{T}} \in \mathbb{R}^{dM}.$$

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

Next we construct the shared iterative layer $\mathcal{J}$, by setting

$$V := \begin{bmatrix} \tilde{I}_{(\tilde{d}-1)M} B/M \\ 0 \end{bmatrix} \begin{bmatrix} LS & \mathbf{0} \\ \mathbf{0} & LI_M \end{bmatrix}, \widetilde{V} := \mathbf{1}_{[M,M]} \otimes V, \text{ and } \boldsymbol{C} := \mathbf{1}_{[M,1]} \otimes \begin{bmatrix} L\tilde{I}_{(\tilde{d}-1)M} A \\ \mathbf{0} \end{bmatrix}.$$

Note that $\widetilde{V}$ is independent of η, and falls into the formulation of V, 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

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

where $\boldsymbol{H}(l) = \left[\widehat{\boldsymbol{h}}_1^l, \widehat{\boldsymbol{h}}_2^l, \ldots, \widehat{\boldsymbol{h}}_{2M-1}^l, \widehat{\boldsymbol{h}}_{2M}^l\right]^{\mathrm{T}}$ 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, $\widehat{\boldsymbol{h}}_{2j}^l \in \mathbb{R}^M$, satisfies

$$\widehat{\boldsymbol{h}}_{2j}^{l+1} = \widehat{\boldsymbol{h}}_{2j}^{l} = \cdots = \widehat{\boldsymbol{h}}_{2j}^{0} = \widetilde{\boldsymbol{G}}^{\eta}, \ \forall l = 0, \cdots, L-1, \forall j = 1, \cdots, M$$

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

$$\widehat{\boldsymbol{h}}_{2j-1}^{l+1} = \widehat{\boldsymbol{h}}_{2j-1}^{l} + S^{+} S\sigma\left(B\left[S\widehat{\boldsymbol{h}}_{2j-1}^{l}, \widetilde{\boldsymbol{G}}^{\eta}\right]^{\mathrm{T}} + A\right), \ \forall l = 0, \cdots, L-1, \forall j = 1, \cdots, M,$$

and

$$\widehat{\boldsymbol{h}}_{1}^{l+1} = \widehat{\boldsymbol{h}}_{3}^{l+1} = \cdots = \widehat{\boldsymbol{h}}_{2M-1}^{l+1}.$$

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

$$\begin{aligned}
&\|\boldsymbol{U}^{\eta} - \boldsymbol{U}^{\eta,*}\| \leq \|\boldsymbol{U}^{\eta} - \boldsymbol{U}^{L}\|_{l^2(\mathbb{R}^M)} + \|\boldsymbol{U}^{L} - \boldsymbol{U}^{\eta,*}\|_{l^2(\mathbb{R}^M)} \\
&\leq \left\|S\hat{\boldsymbol{h}}_1^{L} - \boldsymbol{U}^{L}\right\|_{l^2(\mathbb{R}^M)} + \frac{\varepsilon}{2} \ (\textit{by Assumption 2}) \\
&\leq \left\|S\hat{\boldsymbol{h}}_1^{L-1} - \boldsymbol{U}^{L-1}\right\|_{l^2(\mathbb{R}^M)} + \left\|\hat{\mathscr{R}}\left(S\hat{\boldsymbol{h}}_1^{L-1}, \widetilde{\boldsymbol{G}}\right) - \mathscr{R}\left(\boldsymbol{U}^{L-1}, \widetilde{\boldsymbol{G}}\right)\right\|_{l^2(\mathbb{R}^M)} + \frac{\varepsilon}{2} \\
&\leq \left\|S\hat{\boldsymbol{h}}_1^{L-1} - \boldsymbol{U}^{L-1}\right\|_{l^2(\mathbb{R}^M)} + \left\|\hat{\mathscr{R}}\left(S\hat{\boldsymbol{h}}_1^{L-1}, \widetilde{\boldsymbol{G}}b\right) - \mathscr{R}\left(S\hat{\boldsymbol{h}}_1^{L-1}, \widetilde{\boldsymbol{G}}b\right)\right\|_{l^2(\mathbb{R}^M)} \\
&+ \left\|\mathscr{R}\left(S\hat{\boldsymbol{h}}_1^{L-1}, \widetilde{\boldsymbol{G}}b\right) - \mathscr{R}\left(\boldsymbol{U}^{L-1}, \widetilde{\boldsymbol{G}}b\right)\right\|_{l^2(\mathbb{R}^M)} + \frac{\varepsilon}{2} \\
&\leq (1+m)\left\|S\hat{\boldsymbol{h}}_1^{L-1} - \boldsymbol{U}^{L-1}\right\|_{l^2(\mathbb{R}^M)} + \frac{m\varepsilon}{2(1+m)^{L}} + \frac{\varepsilon}{2}\left(\textit{by Lemma 1 and Assumption 1}\right) \\
&\leq \frac{m\varepsilon}{2(1+m)^{L}}\left(1 + (1+m) + (1+m)^2 + \cdots + (1+m)^{L-1}\right) + \frac{\varepsilon}{2} \\
&\leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon.
\end{aligned}$$

4. Empirical Evaluation

In this section, we demonstrate the empirical effectiveness of the proposed MetaNO approach. Specifically, we conduct experiments on a synthetic dataset from a nonlinear PDE solving problem, a benchmark dataset of heterogeneous materials subject to large deformation, and a real-world dataset from biological tissue mechanical testing. We compare the proposed method against competitive GBML methods as well as two non-meta transfer-learning baselines. All of the experiments are implemented using PyTorch with Adam optimizer, and performed on a workstation with two 2.2 GHz 24-core CPUs and eight NVIDIA Tesla T4 GPUs. For a fair comparison, for each algorithm, we tune the hyperparameters, including the learning rate from {0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001}, the decay rate from {0.5, 0.7, 0.9}, the weight decay parameter from {0.01,

0.001, 0.0001, 0.00001, 0.000001}, and the inner loop learning rate for the baseline GBML methods (MAML and ANIL) from {0.01, 0.001, 0.0001, 0.00001, 0.000001}, to minimize the error on a separate validation dataset. In all experiments we decrease the learning rate with a ratio of learning rate decay rate every 100 epochs. A detailed description of each baseline method is provided in Appendix A.

In all experiments, we considered the averaged relative error, $\|\boldsymbol{u}_{i,\,pred} - \boldsymbol{u}_i\|_{L^2(\Omega)} / \|\boldsymbol{u}_i\|_{L^2(\Omega)}$, as the error metric. We repeat each experiment for 5 times, and report the averaged relative errors and their standard errors.

4.1. Example 1: Synthetic Data Sets and Ablation Study

We first consider the PDE-solution-finding problem of the Holzapfel-Gasser-Odgen (HGO) model [56], which describes the deformation of hyperelastic, anisotropic, and fiber-reinforced materials. In this example, the goal is to find its displacement field $\boldsymbol{u}: [0, 1]^2 \to \mathbb{R}^2$ under different boundary loadings. The specimen is assumed to be subject to a uniaxial tension $T_y(\boldsymbol{x})$ on the top edge (see Figure 2(a)). Therefore, we take the input function $\boldsymbol{g}(\boldsymbol{x})$ as the padded traction loading field, and the output function as the corresponding displacement field.

Data Generation.—To generate training and test samples, the Holzapfel-Gasser-Odgen (HGO) model [56] was employed to describe the constitutive behavior of the material in this example, with its strain energy density function given as:

$$\eta = \frac{E}{4(1+\nu)}(\bar{I}_1 - 2) - \frac{E}{2(1+\nu)}\ln(J) + \frac{k_1}{2k_2}\left(\exp\left(k_2\langle S(\alpha)\rangle^2\right) + \exp\left(k_2\langle S(-\alpha)\rangle^2\right) - 2\right) + \frac{E}{6(1-2\nu)}\left(\frac{J^2-1}{2} - \ln J\right).$$

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

$$S(\alpha) = \frac{\bar{I}_4(\alpha) - 1 + |\bar{I}_4(\alpha) - 1|}{2}.$$

where k_1 and k_2 are fiber modulus and the exponential coefficient, respectively, E is the Young's modulus for the non-fibrous ground matrix, and ν is the Poisson ratio. Moreover, $\bar{I}_1 = \mathrm{tr}(\mathbf{C})$ 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 fiber group with angle direction α from the reference direction, $\bar{I}_4(\alpha) = \mathbf{n}^T(\alpha)\mathbf{C}\mathbf{n}(\alpha)$ is the fourth invariant of the right Cauchy-Green tensor $\mathbf{C}$, where $\mathbf{n}(\alpha) = [\cos(\alpha), \sin(\alpha)]^T$. To generate samples for different specimens, different specimens (tasks) correspond to different material parameter sets, $\{k_1, k_2, E, \nu, \alpha\}$.

To investigate the performance of MetaNO in few-shot learning, we generate 59 training, 1 validation tasks, and 5 in-distribution (ID) test tasks by sampling different physical

parameters k_1, k_2, E, ν, α from the same uniform distribution. To further evaluate the generalizability when the physical parameters of test tasks are outside the training regime, we also generate 2 out-of-distribution (OOD) test tasks with physical parameters from different distributions. For the training tasks, the validation task, and the in-distribution (ID) test task, their physical parameters are sampled from: $k_1, k_2 \sim \mathcal{U}[0.1, 1]$, $E \sim \mathcal{U}[0.55, 1.5]$, $\nu \sim \mathcal{U}[0.01, 0.49]$, and $\alpha \sim \mathcal{U}[\pi/10, \pi/2]$. For the two out-of-distribution (OOD) test tasks, we sample their parameters following $k_1, k_2 \sim \mathcal{U}[1, 1.9]$, $E \sim \mathcal{U}[1.5, 2] \cup \mathcal{U}[0.5, 0.55]$, $\nu \sim \mathcal{U}[0.01, 0.49]$[4], and $\alpha \sim \mathcal{U}[\pi/2, 3\pi/4] \cup [0, \pi/10]$. To generate the high-fidelity (ground-truth) dataset, we sampled 500 different vertical traction conditions $T_y(\boldsymbol{x})$ on the top edge from a random field, following the algorithm in [36, 58]. $T_y(\boldsymbol{x})$ is taken as the restriction of a 2D random field, $\phi(\boldsymbol{x}) = \mathcal{F}^{-1}\left(\gamma^{1/2}\mathcal{F}(\Gamma)\right)(\boldsymbol{x})$, on the top edge. Here, $\Gamma(\boldsymbol{x})$ is a Gaussian white noise random field on $\mathbb{R}^2$, $\gamma = \left(w_1^2 + w_2^2\right)^{-\frac{5}{4}}$ 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 solved the displacement field on the entire domain by minimizing potential energy using the finite element method implemented in FEniCS [59]. 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. The finite element solution was interpolated onto χ, a structured 41 × 41 grid which will be employed as the discretization in our neural operators. The distribution of training and ID/OOD tasks are demonstrated in Figure 3, where one can see that the first OOD task (denoted as "OOD Task1") corresponds to a stiffer material sample and smaller deformation for each given loading, while the second OOD task (denoted as "OOD Task2") generates a softer material sample and larger deformation. For each training task, we generate 500 data pairs $\mathcal{D}^\eta := \{(\boldsymbol{g}_i^\eta, \boldsymbol{u}_i^\eta)\}_{i=1}^{500}$, by sampling the vertical traction loading from a Gaussian random field. Then, the corresponding ground-truth displacement field is obtained using the finite element method implemented in FEniCS [59]. For test tasks, we train with $N^{\text{test}} = \{2, 4, 8, 12, 20, 100, 300\}$ numbers of labelled data pairs (the context set), and evaluate the model on a reserved dataset with 200 data pairs (the target set) on each test task. An 8-layer IFNO is employed as the base model.

Ablation Study.—We first conduct an ablation study on 4 variants of the proposed algorithm: 1) to use the full meta-train and meta-test phases as in Algorithm 1 (denotes as "MetaNO"); 2) to perform steps 1–3 of Algorithm 1, such that only the lifting layer is adapted in the meta-test phase (denotes as "MetaNO-"); 3) to apply task-wise adaptation only to the iterative layers instead of the lift layer in both meta-train and meta-test phases (denoted as "MetaMid"); 4) to apply task-wise adaptation only to the projection layer instead of the lift layer in both meta-train and meta-test phases (denoted as "MetaLast"). We study if the successful "adapting last layers" strategy of MAML and ANIL in image classification problems would apply for our PDE solving problem. Besides these four settings, we also report the few-shot learning results with five baseline methods: 1) Learn a neural operator model only based on the context data set of the test task (denoted as

[4]Here we sample both ID and OOD tasks from the same range of ν, due to the fact that [0.01, 0.49] is the range of Poisson ratio for common materials [57].

"Single"); 2) Pretrain a neural operator model based on all training task data sets, then fine-tune it based on the context test task data set (denoted as "Pretrain1"); 3) Pretrain a single neural operator model based on the context data set of one training task, then fine-tune it based on the context test task data set (denoted as "Pretrain2"); To remove the possible dependency on the pre-training task, in this baseline we randomly select five training tasks for the purpose of pretraining and report the averaged results. 4) MAML, and 5) ANIL. For all experiments we use the full context data set on each training task $(N^{\eta} = 500)$.

As shown in Figure 4, MetaNO- and MetaNO are both able to quickly adapt with few data pairs – to achieve a test error below 5%, "Single" and the two transfer-learning baselines ("Pretrain1", "Pretrain2") require 100+ data pairs, while MetaNO- and MetaNO requires only 4 data pairs. On the other hand, MetaMid, MetaLast, MAML and ANIL have similar performance. They all require 100 data pairs to achieve a < 5% test error. In particular, when using only 4 data pairs on the target task, "MetaNO-" has $2.91\% \pm 0.90\%$ relative test error, while "MetaMid" and "MetaLast" have $58.41\% \pm 9.92\%$ and $43.99\% \pm 11.36\%$ relative test errors, respectively. This observation verifies our finding on the multi-task parametric PDE solution operator learning problem, where one should adapt the first layer, not the middle layers or the last ones. A detailed comparison on the computational time and test errors of these "MetaNO-" and "MetaNO" is provided in Table 1. Herein, the total time of MetaNO- includes the Meta-Train Phase (Steps 1 and 2) and the step of solving for θ_P^{test} in the Meta-Test Phase (Step 3) of Algorithm 1, and the total time of MetaNO includes all steps of both the Meta-Train and Meta-Test Phases (Steps 1–4). Since the Meta-Train Phase took the majority of training time (45612 seconds), although "MetaNO" requires an additional fine-tuning step, its computational cost is of a similar scale as "MetaNO-". When comparing the test errors, we can see that the additional fine-tune step only improves the performance in the larger-sample regime (when $N^{\text{test}} \geq 100$). This fact shows that when given sufficient training context sets, adapting the first layer can capture the underlying task diversity so further fine-tuning may not be needed, especially in the small data regime.

Effect of Varying Training Context Set Sizes.—In this study, we investigate the effect of different training task context sizes $N^{\eta} = \{50, 100, 200, 500\}$ on four meta-learnt models: MetaNO, MetaNO-, MAML, and ANIL. The comparison on test errors are demonstrated in the top plot of Figure 5. Here, MetaNO- and MetaNO did not have any inner loop updates. All parameters from all training tasks are optimized together. In MAML and ANIL we use half of the context set for inner loop updates (support set) and the other half for outer loop updates (target set). One can see that when $N^{\text{test}} \leq 20$, MetaNO- and MetaNO have similar performance and consistently beat MAML and ANIL for both context set sizes. With the increase of N^{test}, the fine-tuning strategy on the test context set becomes more helpful where we see MetaNO becomes more accurate than MetaNO- and MAML beats ANIL. Such effect is more evident on small training context set cases. Perhaps unsurprisingly, with the training task context size varying from 50 to 500, all methods have improved performance with decreasing relative test errors (with the same colors for the same methods across different context dataset). In addition, as the context set size in the test task grows,

fine-tuning will gradually have better performance as MetaNO and MAML beats MetaNO- and ANIL, respectively. Overall, in all combinations of N^{η} and N^{test}, MetaNO achieves the best performance among all models.

In-Distribution and Out-Of-Distribution Tests.—On bottom Figure 5, we demonstrate the relative test error of MetaNO against MAML in both ID and OOD tasks. We can see that test errors of these 3 tasks are in a similar scale as the error on training tasks. The error from OOD task1 is comparable to the averaged ID test task error, while the error from OOD task2 is much larger, as expected by the data property: due to the fact that the solutions in OOD task1 generally have smaller magnitude and hence its solution operator lies more in a linear regime, which makes the solution operator learning task easier. probably due to the fact that the solutions in OOD task1 generally have smaller magnitude and hence its solution operator lies more in a linear regime, which makes the solution operator learning task easier. In all three cases, MetaNO outperforms MAML, hence validating the good generalization performance of MetaNO.

4.2. Example 2: Benchmark Mechanical MNIST Datasets

We further test MetaNO and five baseline methods on benchmark Mechanical MNIST [60]. Mechanical MNIST is a benchmark dataset of heterogeneous material undergoing large deformation, modeled by the Neo-Hookean material with a varying modulus converted from the MNIST bitmap images [60]. It contains 70,000 heterogeneous material specimens, and each specimen is governed by the Neo-Hookean material with a varying modulus converted from the MNIST bitmap images. On each specimen, we have 32 loading/response data pairs on a structured 27 by 27 grid, under the uniaxial extension, shear, equibiaxial extension, and confined compression load scenarios, respectively[5].

Data Generation and Settings.—Here in, we randomly select one specimen corresponding to hand-written number 0 and 2 – 9 respectively as training tasks. Then, among the specimens corresponding to 1, we randomly select six specimens: one for validation and the rest five as the test tasks. Visualization of the ground-truth solutions corresponding to one common loading from different tasks is provided in Figure 6(a), together with the underlying (hidden) microstructure pattern which determines the parameter set $\boldsymbol{b}^{\eta}$. On the meta-train phase, we use the full context data set of all 32 samples for each training task. On the meta-test phase, we reserve 20 data pairs on the test task as the target set for evaluation, then train each model under the few-shot learning setting with $N^{\text{test}} = \{2, 4, 8, 12\}$ labelled data pairs as the context set. All approaches are developed based on an 32-layer IFNO model.

Results.—Besides the diversity of tasks as seen in Figure 6(a), notice that we also have a small number of training tasks ($H = 9$), and a relatively small training context set size ($N^{\eta} = 32$). All these facts make the transfer learning on this benchmark dataset challenging. We present the results in Figure 6(b) and (c). The neural operator model learned by

[5]We have excluded small deformation samples with the maximum displacement magnitude ≤ 0.1.

MetaNO again outperforms the baseline single/transfer learning models and the state-of-the-art GBML models. Our MetaNO model achieves 15% error when using only 2 labelled data pair on the test task, while the Single model has high errors due to overfitting. This fact highlights the importance of learning across multi-tasks: when the total number of measurements on each specimen is limited, it is necessary to transfer the knowledge across specimens. Moreover, while MetaNO-, MAML, and ANIL all have a similar performance in this example, the fine-tuning step in MetaNO seems to substantially improve the accuracy, especially when N^{test} gets larger. This observation is consistent with previous finding on varying training task context sizes.

4.3. Example 3: Application on Real-World Data Sets

We now take a step further to demonstrate the performance of our method on a real-world physical response dataset, which is not generated by solving PDEs. We consider the problem of learning the mechanical response of multiple biological tissue specimens from DIC displacement tracking measurements. We measure the biaxial loading of tricuspid valve anterior leaflet (TVAL) specimens from a porcine heart, such that each specimen (as a task) corresponds to a different region of the leaflet. Due to material heterogeneity of biological tissues, these specimens contain different mechanical and structural properties. However, these properties are generally not observable without damaging the tissue. Therefore, this example provides a proof-of-principle demonstration that how MetaNO applies to learning tasks where the governing equations and physical properties are both unknown, and the dataset has unavoidable measurement noise.

Data Generation.—We now briefly provide the data generation procedure for the tricuspid valve anterior leaflet (TVAL) response modeling example. To generate the data, we firstly followed the established biaxial testing procedure, including acquisition of a healthy porcine heart and retrieval of the TVAL [61, 62]. Then, we sectioned the leaflet tissue and applied a speckling pattern to the tissue surface using an airbrush and black paint [63–65]. The painted specimen was then mounted to a biaxial testing device (BioTester, CellScale, Waterloo, ON, Canada). To generate samples for each specimen, we performed 7 protocols of displacement-controlled testing to target various biaxial stresses: $P_{11}:P_{22}=\{1:1,1:0.66,1:0.33,0.66:1,0.33:1,0.05:1,1:0.1\}$. Here, P_{11} and P_{22} denote the first Piola-Kirchhoff stresses in the x- and y-directions, respectively. Each stress ratio was performed for three loading/unloading cycles. Throughout the test, images of the specimen were captured by a CCD camera, and the load cell readings and actuator displacements were recorded at 5 Hz. After testing, the acquired images were analyzed using the digital image correlation (DIC) module of the BioTester's software. The pixel coordinate locations of the DIC-tracked grid were then exported and extrapolated to a 21 by 21 uniform grid. In this example, we have the DIC measurements on 16 specimens, with 500 data pairs of loadings and material responses from the 7 protocols on each specimen. These specimens are divided into three groups: 12 for the purpose of meta-train, 1 for validation, and 3 for test. This reflects a common challenge in scientific applications, we not only have limited samples per task, the number of available training tasks is also limited. To demonstrate the diversity of these specimens due to the material heterogeneity in biological tissues, in Figure 7 we plot

the processed displacement field of two exemplar training specimens and the validation and test specimens. For each model, the results are reported as the average of all 3 test tasks.

Results.—Herein, we aim to model the tissue response by learning a neural operator mapping the boundary displacement loading to the interior displacement field on each tissue specimen. We use 13 specimens for training and validation with context size $N^{\eta} = 500$, and provide the test results as the average on the rest 3 specimens. With a 4-layer IFNO as the base model, we train each model based on $N^{\text{test}} \in [2, 300]$ samples, and then evaluate the performance on another 200 samples. The results are provided in Figure 8. MetaNO performs the best among all the methods across all N^{test}, beating MAML and ANIL by a significant margin. Interestingly, MAML and ANIL did not even beat the "Pretrain1" method, possibly due to the low efficacy of the adapting last layers strategy and the small number of training tasks.

5. Summary and Future Directions

In this paper we propose MetaNO, the first neural-operator-based meta-learning approach that is designed to achieve good transferability in learning complex physical system responses. In particular, we focus on the scenario that the physical responses of different systems can be modeled by a common parametric PDE, and the behavior of each system (specimen) is governed by a different underlying parameter field. Then, based on the observed data from several known systems, the goal is to efficiently learn a surrogate model, which acts as the surrogate solution operator for the hidden PDE, for a new and unseen system. Comparing with existed transfer-learning techniques, our MetaNO features a novel *first layer adaption architecture*, which is theoretically motivated and shown to be a universal solution operator for multiple parametric PDE solving tasks. We demonstrate the effectiveness of our proposed MetaNO algorithm on various synthetic, benchmark, and real-world datasets, showing promises with significant improvement in sample efficiency over baseline methods.

Although in this paper we focus on material modeling problems in experiments, the proposed approach is a machine learning model that can handle different PDEs with heterogeneous coefficients, which is common in physics problems and not limited in material modeling. For example, another suitable example is climate modeling, where cities responses, temperature exposures, and impacts are all heterogeneous, resulting in different parameters in the PDE model. Here we choose material as the application mainly due to the availability of data. For future work, we will investigate the applicability of the proposed approach to other scientific domains. Moreover, we point out that the universal approximator property of MetaNO relies on the assumption that all systems are governed by a common parametric PDE. As such, a set of common characteristic solution operator features can be obtained from the training tasks in the form of the common layer parameters, and then can be utilized to accelerate the learning efficacy in the test task. Therefore, we anticipate the method to suffer from performance deterioration when handling the transferability between different types of PDEs, e.g., when all training tasks are solving diffusion problems but the test task is on an advection problem. It would be an interesting future direction of extending

MetaNO to handle such a challenging scenario. In this paper, we employed the implicit Fourier neural operator [39] as the base model. As another natural extension, one might consider the proposed meta-learning approach based on other types of neural operators and neural networks, such as the graph neural operators [26, 29] and the general graph neural networks [66, 67]. As such, the model will be capable to handle measurements from unstructured grid points and domains, which be valuable for applications with complex and possibly evolving domain geometries.

Acknowledgements

L. Zhang, H. You and Y. Yu would like to acknowledge support by the National Science Foundation under award DMS-1753031 and the AFOSR grant FA9550-22-1-0197. C.-H. Lee would like to thank the support from the National Institutes of Health (NIH) Grant R01 HL159475 and the Presbyterian Health Foundation Team Science Grants. Portions of this research were conducted on Lehigh University's Research Computing infrastructure partially supported by NSF Award 2019035.

Appendix A.: Formulation of Baseline Methods

In this section, we discuss each baseline methods in details and how they are used in our experiments. A meta-learning baseline in our problem setting would be to apply MAML and ANIL to a neural operator architecture. Here we formally state the implementation of ANIL and MAML for the problem described above, and they will serve as the baselinebaseline meta-based methods in our empirical experiments.

MAML.

The MAML algorithm proposed in [5] aims to find an initialization, $\tilde{\theta}$, across all tasks, so that new tasks can be learnt with very few gradient updates and examples. First, a batch $\{\mathcal{T}^\eta\}_{\eta=1}^{H}$ of H tasks are drawn from the training task set. For each task $\mathcal{T}^\eta$, the context set of loading field/response field data pairs $\mathcal{D}^\eta$ is split to a support set of samples, $\mathcal{S}^\eta$, which will be used for inner loop updates, and a target set of samples, $\mathcal{Z}^\eta$, for outer loop updates. Then, for the inner loop, let $\theta^{\eta,0} := \tilde{\theta}$ and $\theta^{\eta,i}$ be the task-wise parameter after i-th gradient update. During each inner loop update, the task-wise parameter is updated via

$$\theta^{\eta,i} = \theta^{\eta,i-1} - \alpha \nabla_{\theta^{\eta,i-1}} \mathcal{L}_{\mathcal{S}^\eta}\left(\theta^{\eta,i-1}\right), \text{ for } \eta = 1, \cdots, H, \tag{A.1}$$

where $\mathcal{L}_{\mathcal{S}^\eta}\left(\theta^{\eta,i-1}\right)$ is the loss on the support set of the η-th task, and α is the step size. After m inner loop updates, the initial parameter $\tilde{\theta}$ is updated with a fixed step size β:

$$\tilde{\theta} \leftarrow \tilde{\theta} - \beta \nabla_{\tilde{\theta}} \mathcal{L}_{\text{meta}}\left(\tilde{\theta}\right), \text{ where the meta-loss } \mathcal{L}_{\text{meta}}\left(\tilde{\theta}\right) := \sum_{\eta=1}^{H} \mathcal{L}_{\mathcal{Z}^\eta}\left(\theta^{\eta,m}\right). \tag{A.2}$$

Then, on the test task, $\mathcal{T}^{\text{test}}$, an inner loop adaptation is performed based on few labelled samples $\mathcal{D}^{\text{test}}$ until convergence, and the approximated solution operator model is obtained on the test task as $\widetilde{\mathcal{G}}\left[\boldsymbol{g}; \theta^{\text{test}}\right]$.

ANIL.

In [10], ANIL was proposed as a modified version of MAML with inner loop updates only for the final layer. The inner loop update formulation of (A.1) is modified as

$$\theta_Q^{\eta,i} = \theta_Q^{\eta,i-1} - \alpha \nabla_{\theta_Q^{\eta,i-1}} \mathcal{L}_{s^\eta}\left(\theta_Q^{\eta,i-1}\right), \text{ for } \eta = 1, \cdots, H, \tag{A.3}$$

where $\theta_Q^{\eta,i}$ is the task-wise parameter on the final (projection) layer after ith gradient update. Then, the same outer loop updates are performed following (A.2).

Single/Pretrain1/Pretrain2.

We also implemented 3 non-meta-learning baseline approaches.

- **Single**: Learn a neural operator model only based on the context data set of the test task.
- **Pretrain1**: Pretrain a neural operator model based on all training task data sets, then fine-tune it based on the context test task data set.
- **Pretrain2**: Pretrain a single neural operator model based on the context data set of one training task, then fine-tune it based on the context test task data set. To remove the possible dependency on the pre-training task, in this baseline we randomly select five training tasks for the purpose of pretraining and report the averaged results.

Appendix B.: Training Details

In the following we briefly describe the hyperparameter settings employed in running of each algorithm. The code and the processed datasets will be publicly released at Github for readers to reproduce the experimental results.

Appendix B.1. Example 1: Synthetic Data Sets

Base model:

As the base model for all algorithms, we construct an architecture for IFNO [39] as follows. First, the input loading field instance $\boldsymbol{g}(\boldsymbol{x}) \in \mathcal{A}$ is lifted to a higher dimensional representation via lift layer $\mathcal{P}[\boldsymbol{g}](\boldsymbol{x})$, which is parameterized as a 1-layer feed forward linear layer with width $(3, 32)$. Then for the iterative layer in (2.1), we implement $\mathcal{F}^{-1}[\mathcal{F}[\kappa(\cdot; \boldsymbol{v})] \cdot \mathcal{F}[\boldsymbol{h}(\cdot, l)]](\boldsymbol{x})$ with 2D fast Fourier transform (FFT) with input channel and output channel widths both set as 32 and the truncated Fourier modes set as 8. The local linear transformation parameter, W, is parameterized as a 1-layer feed forward network with

width (32,32). In the projection layer, a 2-layer feed forward network with width (32,128,2) is employed. To accelerate the training procedure, we apply the shallow-to-deep training technique to initialize the optimization problem. In particular, we start from the NN model with depth $L = 1$, train until the loss function reaches a plateau, then use the resultant parameters to initialize the parameters for the next depth, with $L = 2$, $L = 4$, and $L = 8$. In the synthetic experiments, we set the layer depth as $L = 8$.

MetaNO:

During the meta-train phase, we train for the task-wise parameters θ_P^n and the common parameters θ_I and θ_Q on all 59 training tasks, with the context set of 500 samples on each task. After meta-train phase, we load θ_I and θ_Q and the averaged θ_P^n among all 59 tasks as initialization, then tune the hyperparameters based on the validation task. In particular, the 500 samples on the validation task is split into two parts: 300 samples are reserved for the purpose of training (as the context set) and the rest 200 samples are used for evaluation (as the target set). Then we train for the lift layer on the validation task, and tune the learning rate, the decay rate, and the weight decay parameter for different context set sizes $\left(N^{\text{test}}\right)$, to minimize the loss on the target set. Based on the chosen hyperparameters, we perform the test on the test task by training for the lift layer on different numbers of samples on its context set, then evaluate and report the performance based on its target set. We repeat the procedure on the test task with selected hyperparameters with different 5 random seeds, and calculate means and standard errors for the resultant test errors on target set.

MAML&ANIL:

For MAML and ANIL, we use the same architecture as the base model, and also split the training tasks for the purpose of training (59 tasks) and validation (1 task) as in MetaNO. During the meta-train phase, for each task we randomly split the available 500 samples to two sets: 250 samples in the support set used for inner loop updates, and the rest in the target set for outer loop updates. During the inner loop update, we train for the task-wise parameter with one epoch, following the standard settings of MAML and ANIL [5, 10]. Then, the model hyperparameters, including the learning rate, weight decay, decay rate, and inner loop learning rate, are tuned. In the meta-test phase, we load the initial parameter and train for all parameters (in MAML) or the last-layer parameters (in ANIL) until the optimization algorithm converges. Similar as in MetaNO, we first tune the hyperparameters on the validation task, then evaluate the performance on the test task.

Appendix B.2. Example 2: Mechanical MNIST

Base model:

As the base model for all algorithms, we construct two IFNO architectures, for the prediction of u_x and u_y, the displacement fields in the x- and y-directions, respectively. On each architecture, the input loading field instance $\boldsymbol{g}(\boldsymbol{x}) \in \mathscr{A}$ is mapped to a higher dimensional representation via a lifting layer $\mathscr{P}[\boldsymbol{g}](\boldsymbol{x})$ parameterized as a 1-layer feed forward linear layer with width (4,64). Then for the iterative layer in (2.1), we set the number of truncated Fourier mode as 13, and parameterize the local linear transformation

parameter, W, as a 1-layer feed forward network with width (64,64). In the projection layer, a 2-layer feed forward network with width (64,128,1) is employed. In this example we also apply the shallow-to-deep technique to accelerate the training, and set the layer depth as $L = 32$.

MetaNO:

During the meta-train phase, we train for the task-wise parameters θ_P^η and the common parameters θ_I and θ_Q on all 9 training tasks, with the context set of 32 samples on each task. After the meta-train phase, we load θ_I and θ_Q and the averaged θ_P^η among all 9 tasks as initialization, then train θ_P on the validation task. In particular, the 32 samples on the validation task is split into two parts: 12 samples are reserved for the purpose of training (as the context set) and the rest 20 samples are used for the purpose of evaluation (as the target set). Then we train the lift layer on the validation task, and tune the learning rate, the decay rate, and the weight decay parameter for different context set sizes $\left(N^{\text{test}}\right)$, to minimize the loss on the target set. Based on the chosen hyperparameters, we perform the meta-test phase on the test task by training for the lift layer on different numbers of samples on its context set, then evaluate and report the performance based on its target set. We repeat the procedure with different 5 random seeds on each of the 5 test tasks, and calculate means and standard errors for the resultant test errors on the target set.

MAML&ANIL:

For MAML and ANIL, we use the same architecture as the base model. During the meta-train phase, for each task we randomly split the available 32 samples to two sets: 16 samples in the support set used for inner loop updates, and the rest in the target set for outer loop updates. During the inner loop update, we also follow the standard settings of MAML and ANIL [5, 10], and tune the hyperparameters following the same procedure as elaborated above for Example 1.

Appendix B.3. Example 3: Experimental Measurements on Biological Tissues

Base model:

As the base model, we first construct the lifting layer as a 1-layer feed forward linear layer with width (4,16). Then for the iterative layer in we keep 8 truncated Fourier modes and parameterize the local linear transformation parameter, W, a 1-layer feed forward network with width (16,16). In the projection layer, a 2-layer feed forward network with width (16,64,1) is employed. We construct two 4-layer IFNO architectures, for the prediction of u_x and u_y, the displacement fields in the x - and y-directions, respectively.

MetaNO:

During the meta-train phase, we train for the task-wise parameters θ_P^η and the common parameters θ_I and θ_Q on all 12 tasks, with the context set of 500 samples on each task. After meta-train phase, we load θ_I and θ_Q and the averaged θ_P^η among all 12 tasks as initialization,

then tune the hyperparameters based on the validation task. In particular, the 500 samples on the validation task is divided into two parts: 300 samples are reserved for the purpose of training (as the context set) and the rest 200 samples are used for evaluation (as the target set). Based on the chosen hyperparameters, we perform the test on the test tasks by training for the lift layer on different numbers of samples on its context set, then evaluate the performance based on its target set.

MAML&ANIL:

For MAML and ANIL, we use the same architecture as the base model, and also split the training tasks for the purpose of training and validation as in MetaNO. During the meta-train phase, for each task we randomly split the available 500 samples to two sets: 250 samples in the support set used for inner loop updates, and the rest in the target set for outer loop updates. During the inner loop update, we train for the task-wise parameter with one epoch, following the standard settings of MAML and ANIL [5,10].

References

[1]. Wang Y, Yao Q, Kwok JT, Ni LM, Generalizing from a few examples: A survey on few-shot learning, ACM computing surveys (csur) 53 (3) (2020) 1–34.

[2]. Koch G, Zemel R, Salakhutdinov R, et al., Siamese neural networks for one-shot image recognition, in: ICML deep learning workshop, Vol. 2, Lille, 2015, p. 0.

[3]. Vinyals O, Blundell C, Lillicrap T, Wierstra D, et al. , Matching networks for one shot learning, Advances in neural information processing systems 29.

[4]. Snell J, Swersky K, Zemel R, Prototypical networks for few-shot learning, Advances in neural information processing systems 30.

[5]. Finn C, Abbeel P, Levine S, Model-agnostic meta-learning for fast adaptation of deep networks, in: International Conference on Machine Learning, Vol. 70, PMLR, 2017, pp. 1126–1135.

[6]. Santoro A, Bartunov S, Botvinick M, Wierstra D, Lillicrap T, Meta-learning with memory-augmented neural networks, in: International conference on machine learning, PMLR, 2016, pp. 1842–1850.

[7]. Antoniou A, Edwards H, Storkey A, How to train your maml, arXiv preprint arXiv:1810.09502.

[8]. Ravi S, Larochelle H, Optimization as a model for few-shot learning.

[9]. Nichol A, Schulman J, Reptile: a scalable metalearning algorithm, arXiv preprint arXiv:1803.02999 2 (3) (2018) 4

[10]. Raghu A, Raghu M, Bengio S, Vinyals O, Rapid learning or feature reuse? towards understanding the effectiveness of maml, arXiv preprint arXiv:1909.09157.

[11]. Tripuraneni N, Jin C, Jordan M, Provable meta-learning of linear representations, in: International Conference on Machine Learning, PMLR, 2021, pp. 10434–10443.

[12]. Collins L, Mokhtari A, Oh S, Shakkottai S, Maml and anil provably learn representations, arXiv preprint arXiv:2202.03483.

[13]. Zhang W, Sacks MS, Modeling the response of exogenously crosslinked tissue to cyclic loading: The effects of permanent set, Journal of the Mechanical Behavior of Biomedical Materials 75 (2017) 336–350. [PubMed: 28780254]

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

[15]. Rieppo J, Hallikainen J, Jurvelin JS, Kiviranta I, Helminen HJ, Hyttinen MM, 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. [PubMed: 18072283]

[16]. Xu J, Ton J-F, Kim H, Kosiorek A, Teh YW, Metafun: Meta-learning with iterative functional updates, in: International Conference on Machine Learning, PMLR, 2020, pp. 10617–10627.

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

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

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

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

[21]. Zhang L, Han J, Wang H, Car R, Weinan E, Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics, Physical Review Letters 120 (14) (2018) 143001. [PubMed: 29694129]

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

[23]. Pfau D, Spencer JS, Matthews AG, Foulkes WMC, Ab initio solution of the many-electron schrödinger equation with deep neural networks, Physical Review Research 2 (3) (2020) 033429.

[24]. He Q, Laurence DW, Lee C-H, Chen J-S, Manifold learning based data-driven modeling for soft biological tissues, Journal of Biomechanics 117 (2021) 110124. [PubMed: 33515902]

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

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

[27]. Li Z, Kovachki N, Azizzadenesheli K, Liu B, Stuart A, Bhattacharya K, Anandkumar A, Multipole graph neural operator for parametric partial differential equations, Advances in Neural Information Processing Systems 33 (2020) NeurIPS 2020.

[28]. Li Z, Kovachki NB, Azizzadenesheli K, Bhattacharya K, Stuart A, Anandkumar A, Fourier NeuralOperator for Parametric Partial Differential Equations, in: International Conference on Learning Representations, 2020.

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

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

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

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

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

[34]. Goswami S, Bora A, Yu Y, Karniadakis GE, Physics-informed neural operators, 2022 arXiv preprint arXiv:2207.05748

[35]. Yin M, Ban E, Rego BV, Zhang E, Cavinato C, Humphrey JD, Em Karniadakis G, Simulating progressive intramural damage leading to aortic dissection using DeepONet: an operator-regression neural network, Journal of the Royal Society Interface 19 (187) (2022) 20210670. [PubMed: 35135299]

[36]. Yin M, Zhang E, Yu Y, Karniadakis GE, Interfacing finite elements with deep neural operators for fast multiscale modeling of mechanics problems, Computer Methods in Applied Mechanics and Engineering, in press (2022) 115027. [PubMed: 37384215]

[37]. Lu L, He H, Kasimbeg P, Ranade R, Pathak J, One-shot learning for solution operators of partial differential equations, arXiv preprint arXiv:2104.05512.

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

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

[40]. Pan SJ, Yang Q, A survey on transfer learning, IEEE Transactions on knowledge and data engineering 22 (10) (2009) 1345–1359.

[41]. Thrun S, Pratt L, Learning to learn: Introduction and overview, in: Learning to learn, Springer, 1998, pp. 3–17.

[42]. Hospedales T, Antoniou A, Micaelli P, Storkey A, Meta-learning in neural networks: A survey, 2020 arXiv preprint arXiv:2004.05439.

[43]. Mai H, Le TC, Hisatomi T, Chen D, Domen K, Winkler DA, Caruso RA, Use of meta models for rapid discovery of narrow bandgap oxide photocatalysts, iScience (2021) 103068. [PubMed: 34585115]

[44]. Zhang L, You H, Yu Y, Metanor: A meta-learnt nonlocal operator regression approach for metamaterial modeling, arXiv preprint arXiv:2206.02040.

[45]. Yin Y, Ayed I, de Bézenac E, Baskiotis N, Gallinari P, Leads: Learning dynamical systems that generalize across environments, Advances in Neural Information Processing Systems 34 (2021) 7561–7573.

[46]. Wang R, Walters R, Yu R, Meta-learning dynamics forecasting using task inference, arXiv preprint arXiv:2102.10271

[47]. Kailkhura B, Gallagher B, Kim S, Hiszpanski A, Han TY-J, Reliable and explainable machine-learning methods for accelerated material discovery, npj Computational Materials 5 (1) (2019) 1–9.

[48]. Goswami S, Kontolati K, Shields MD, Karniadakis GE, Deep transfer learning for partial differential equations under conditional shift with deeponet, arXiv preprint arXiv:2204.09810.

[49]. Yoon J, Kim T, Dia O, Kim S, Bengio Y, Ahn S, Bayesian model-agnostic meta-learning, in: Proceedings of the 32nd International Conference on Neural Information Processing Systems (NeurIPS 2018), 2018, pp. 7343–7353.

[50]. Vanschoren J, Meta-learning: A survey, 2018 arXiv preprint arXiv:1810.03548.

[51]. Yang H, Kwok J, Efficient variance reduction for meta-learning, in: International Conference on Machine Learning, PMLR, 2022, pp. 25070–25095.

[52]. Kalais K, Chatzis S, Stochastic deep networks with linear competing units for model-agnostic meta-learning, in: International Conference on Machine Learning, PMLR, 2022, pp. 10586–10597.

[53]. Dejam M, Hassanzadeh H, Chen Z, Pre-darcy flow in porous media, Water Resources Research 53 (10) (2017) 8187–8210.

[54]. Fallah F, Ghajari M, Safa Y, Computational modelling of dynamic delamination in morphing composite blades and wings, The International Journal of Multiphysics 13 (4) (2019) 393–430.

[55]. Wei C, Wojnar CS, Wu C, Hydro-chemo-mechanical phase field formulation for corrosion induced cracking in reinforced concrete, Cement and Concrete Research 144 (2021) 106404.

[56]. Holzapfel GA, Gasser TC, Ogden RW, 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.

[57]. Bischofs IB, Schwarz US, Effect of poisson ratio on cellular structure formation, Physical review letters 95 (6) (2005) 068102. [PubMed: 16090996]

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

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

[60]. Lejeune E, Mechanical mnist: A benchmark dataset for mechanical metamodels, Extreme Mechanics Letters 36 (2020) 100659.

[61]. Ross CJ, Laurence DW, Richardson J, Babu AR, Evans LE, Beyer EG, Childers RC, Wu Y, Towner RA, Fung K-M, Mir A, Burkhart HM, Holzapfel GA, Lee C-H, An investigation of the glycosaminoglycan contribution to biaxial mechanical behaviours of porcine atrioventricular heart valve leaflets, Journal of The Royal Society Interface 16 (156) (2019) 20190069. [PubMed: 31266416]

[62]. Laurence D, Ross C, Jett S, Johns C, Echols A, Baumwart R, Towner R, Liao J, Bajona P, Wu Y, et al. , An investigation of regional variations in the biaxial mechanical properties and stress relaxation behaviors of porcine atrioventricular heart valve leaflets, Journal of Biomechanics 83 (2019) 16–27. [PubMed: 30497683]

[63]. Zhang DS, Arola DD, Applications of digital image correlation to biological tissues, Journal of Biomedical Optics 9 (4) (2004) 691–699. [PubMed: 15250755]

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

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

[66]. Li A, Zhang YJ, Isogeometric analysis-based physics-informed graph neural network for studying traffic jam in neurons, Computer Methods in Applied Mechanics and Engineering 403 (2023) 115757.

[67]. Li A, Barati Farimani A, Zhang YJ, Deep learning of material transport in complex neurite networks, Scientific reports 11 (1) (2021) 11280. [PubMed: 34050208]

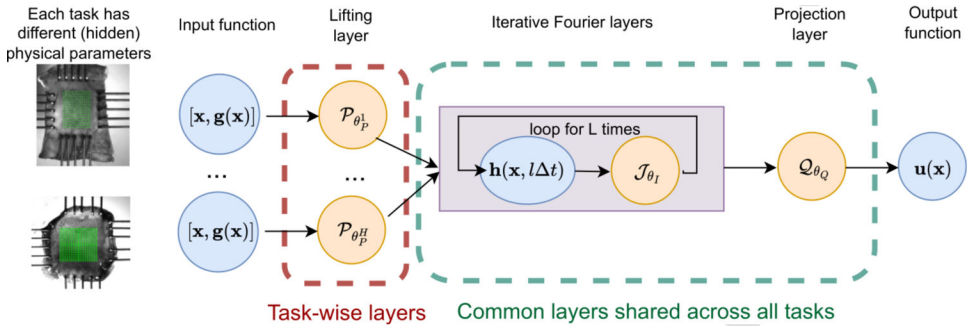

Figure 1:
The architecture of MetaNO based on an integral neural operator model.

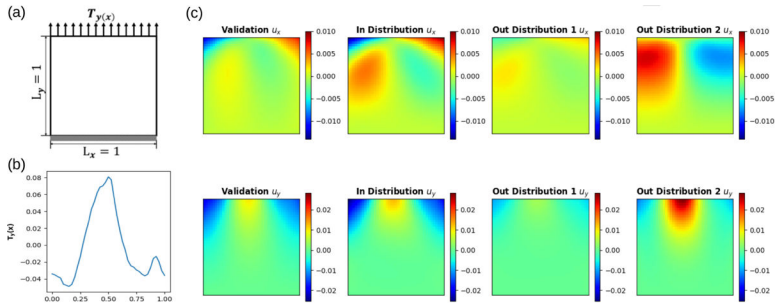

Figure 2:

Problem setup of example 1: the synthetic data sets. (a) A unit square specimen subject to uniaxial tension with Neumann-type boundary condition. (b) & (c) Visualization of an instances of the loading field $T_y(x)$, and the corresponding ground-truth solutions $\boldsymbol{u}^{\eta}(\boldsymbol{x})$ from the in-distribution and out-of-distribution tasks, showing the solution diversity across different tasks, due to the change of underlying hidden material parameter set. □

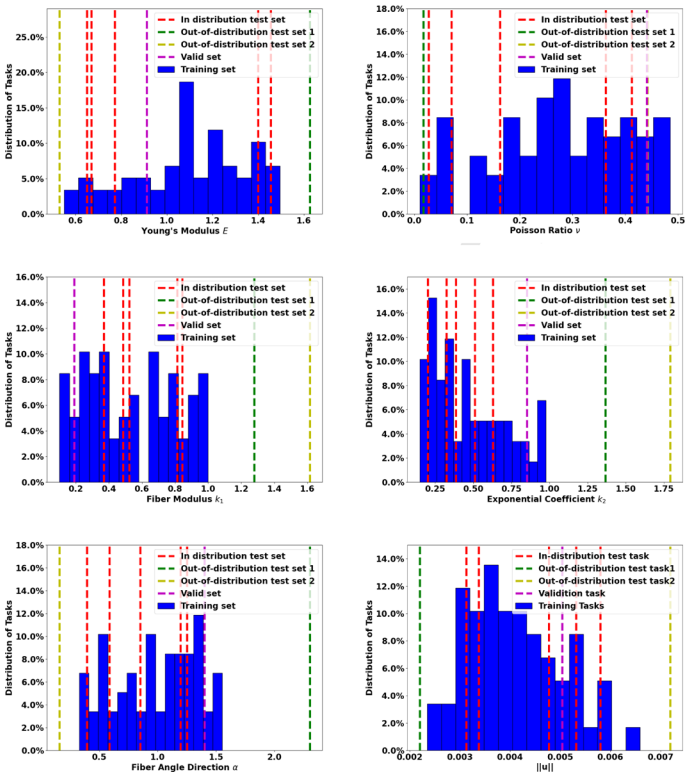

Figure 3:

Distribution of physical parameters of different tasks in example 1, and the resultant magnitude of material response, $\|u^{\eta}(\boldsymbol{x})\|_{L^2(\Omega)}$, on an exemplar loading instance shown in Figure 2(b).

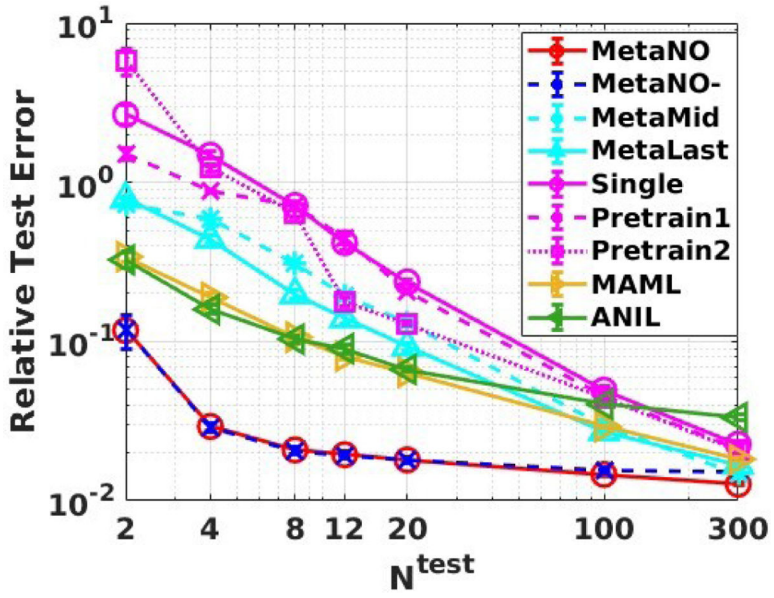

Figure 4:

Results of the ablation study on example 1: the synthetic data set. Comparison on test errors in the in-distribution test, when using the full context set $(N^{\eta} = 500)$ on training tasks and different sizes of context set (N^{test}) on test tasks.

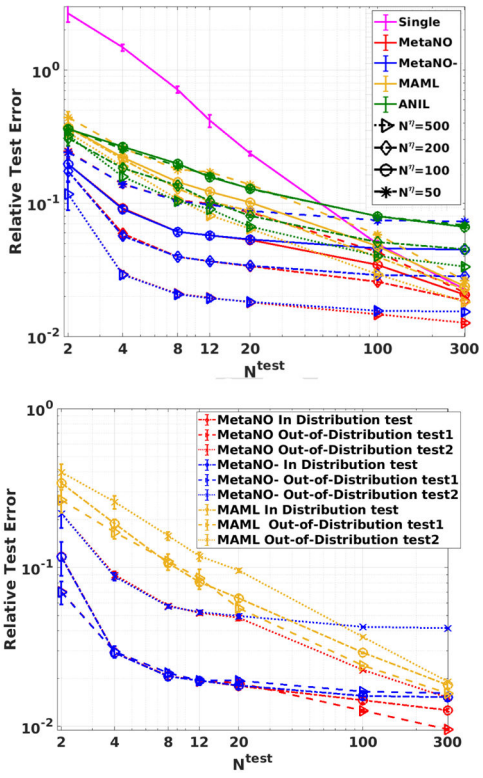

Figure 5:

Results on on example 1: the synthetic data set. Top: The relative test error results showing the effect of varying training task context set sizes $N^{\eta} \in \{50, 100, 200, 500\}$. Bottom: The relative error of MetaNO and MAML in in-distribution and out-of distribution tests.

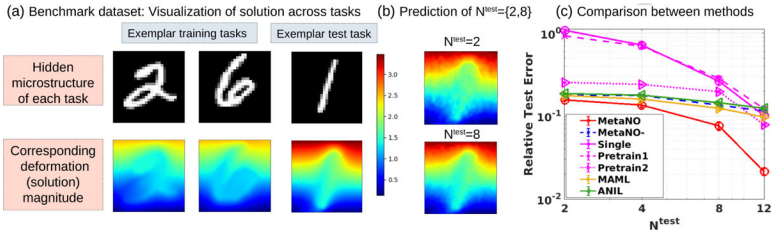

Figure 6:

Results on example 2: the benchmark (Mechanical MNIST [60]) dataset. (a) The visualization of different tasks, their underlying microstructure field $\boldsymbol{b}^{\eta}$, and the corresponding ground-truth solution. (b) Prediction results based on few samples $\left(N^{\text{test}} = 2\right.$ and $\left. N^{\text{test}} = 8\right)$ on a test task. (c) Comparison of MetaNO and five baseline methods.

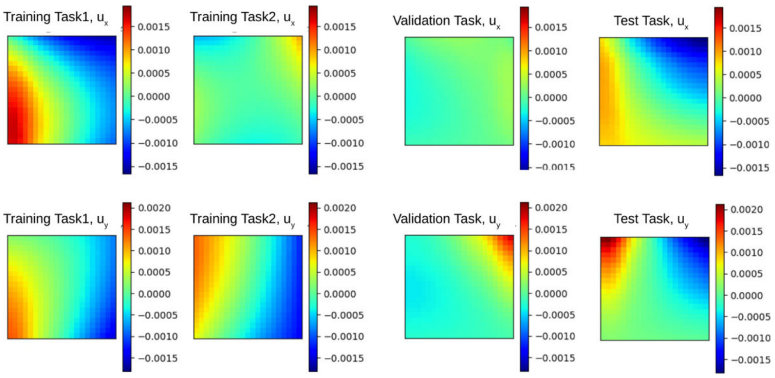

Figure 7:

Visualization of the processed dataset in example 3: learning the biological tissue responses. Subject to the same loading instance, different columns show the corresponding ground-truth solutions $\boldsymbol{u}^{\eta}(\boldsymbol{x})$ from different tasks, showing the solution diversity across different tasks due to the change of underlying hidden material parameter field.

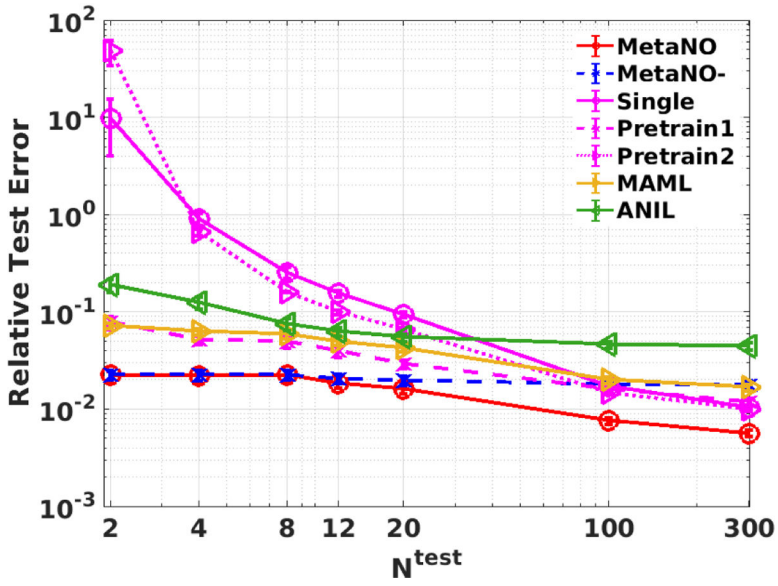

Figure 8:

Comparison of MetaNO and five baseline methods in example 3: learning the biological tissue responses. This example features measurement noise and a small number of available tasks.

Table 1:

A comparison of the computational costs (in seconds) and test errors (in relative L^2 norm) of MetaNO and MetaNO-, in the ablation study on example 1. Here, the total time of MetaNO- includes the Meta-Train Phase (Steps 1 and 2) and the step of solving for θ_P^{test} in the Meta-Test Phase (Step 3), and the total time of MetaNO includes all steps of both the Meta-Train and Meta-Test Phases (Steps 1–4).

			MetaNO-		MetaNO	
N^{test}	Step 3	Step 4	Total Time	Test Error	Total Time	Test Error
2	24	25	45636	11.70%±14.01%	45661	11.72%±14.02%
4	48	49	45660	2.91%±0.90%	45709	2.93%±0.90%
8	93	98	45705	2.07%±0.53%	45803	2.08%±0.53%
12	141	147	45753	1.94%±0.56%	45900	1.92%±0.55%
20	223	247	45835	1.82%±0.41%	46082	1.80%±0.39%
100	976	1025	46588	1.56%±0.35%	47613	1.46%±0.26%
300	2116	2282	47728	1.53%±0.34%	50010	1.27%±0.15%