Check for updates

International Journal for Numerical Methods in Engineering

WILEY

RESEARCH ARTICLE OPEN ACCESS

A Finite Operator Learning Technique for Mapping the Elastic Properties of Microstructures to Their Mechanical Deformations

Shahed Rezaei[1] | Reza Najian Asl[2] | Shirko Faroughi[3] | Mahdi Asgharzadeh[3] | Ali Harandi[4] | Rasoul Najafi Koopas[5] | Gottfried Laschet[1] | Stefanie Reese[4] | Markus Apel[1]

[1]ACCESS e.V., Aachen, Germany | [2]Chair of Structural Analysis, Technical University of Munich, München, Germany | [3]Department of Mechanical Engineering, Urmia University of Technology, Urmia, Iran | [4]Institute of Applied Mechanics, RWTH Aachen University, Aachen, Germany | [5]Institute of Solid Mechanics, Helmut-Schmidt University/University of the Federal Armed Forces, Hamburg, Germany

Correspondence: Shahed Rezaei (s.rezaei@access-technology.de) | Shirko Faroughi (sh.farughi@uut.ac.ir)

Received: 1 June 2024 | **Revised:** 19 September 2024 | **Accepted:** 19 November 2024

Funding: This research was supported by Deutsche Forschungsgemeinschaft (DFG) (project: 390621612).

Keywords: Fourier-based parametrization | microstructure | operator learning | physics-informed neural networks

ABSTRACT

To obtain fast solutions for governing physical equations in solid mechanics, we introduce a method that integrates the core ideas of the finite element method with physics-informed neural networks and concept of neural operators. We propose directly utilizing the available discretized weak form in finite element packages to construct the loss functions algebraically, thereby demonstrating the ability to find solutions even in the presence of sharp discontinuities. Our focus is on micromechanics as an example, where knowledge of deformation and stress fields for a given heterogeneous microstructure is crucial for further design applications. The primary parameter under investigation is the Young's modulus distribution within the heterogeneous solid system. Our investigations reveal that physics-based training yields higher accuracy compared with purely data-driven approaches for unseen microstructures. Additionally, we offer two methods to directly improve the process of obtaining high-resolution solutions, avoiding the need to use basic interpolation techniques. The first one is based on an autoencoder approach to enhance the efficiency for calculation on high resolution grid points. Next, Fourier-based parametrization is utilized to address complex 2D and 3D problems in micromechanics. The latter idea aims to represent complex microstructures efficiently using Fourier coefficients. The proposed approach draws from finite element and deep energy methods but generalizes and enhances them by learning parametric solutions without relying on external data. Compared with other operator learning frameworks, it leverages finite element domain decomposition in several ways: (1) it uses shape functions to construct derivatives instead of automatic differentiation; (2) it automatically includes node and element connectivity, making the solver flexible for approximating sharp jumps in the solution fields; and (3) it can handle arbitrary complex shapes and directly enforce boundary conditions. We provided some initial comparisons with other well-known operator learning algorithms, further emphasize the advantages of the newly proposed method.

This is an open access article under the terms of the Creative Commons Attribution-NonCommercial License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited and is not used for commercial purposes.

© 2024 The Author(s). *International Journal for Numerical Methods in Engineering* published by John Wiley & Sons Ltd.

1 | Introduction

By employing advanced simulation techniques, researchers can model and analyze the complex interplay of stresses, and deformations, providing thus a deeper comprehension of material properties. These simulations facilitate the exploration of diverse materials and contribute significantly to the design and optimization of materials with tailored mechanical characteristics, ultimately advancing innovation in numerous manufacturing industries. Examples of this include obtaining the mechanical behavior of the microstructure through numerical methods, which then, through homogenization, one obtains the material properties at the macro scale [1].

Notable examples of simulation techniques include finite difference, finite volume, and finite element methods (FEM) [2, 3]. Despite their predictive power, these methods have two major drawbacks. First, obtaining the solution from them can easily become very time-consuming and they are not fast enough for many upcoming design applications. Second, as soon as any parameter changes (e.g., the morphology of the microstructure), one has to recompile the adapted model and redo the computation to obtain the new solution. In other words, the standard solvers are limited to one particular boundary value problem (BVP) and do not seem to be a sustainable and green choice, since they are designed to be used only once.

Deep learning (DL) methods provide solutions to the stated problem by leveraging their robust interpolation capabilities and mapping disparate input and output spaces together. For an overview of the potential of deep learning methods in the field of computational material mechanics see [2, 4, 5]. The interpolation power of deep learning models is raised to a level that encourages researchers to train the neural network to learn the solution to a given BVP in a parametric way. This idea is now pressed as operator learning, which involves the mapping of two infinite spaces or functions to each other. A major advantage is that once the training is done, the network evaluation for obtaining the solution is usually an extremely cheap process in terms of computational cost. Some well-established methods for operator learning include but are not limited to, DeepOnet [6], Fourier Neural Operator [7], Graph Neural Operator [8], and Laplace Neural Operator [9, 10].

1.1 | Data-Driven Operator Learning

In this category, the data for the training is obtained from the available resources or by performing offline computations and/or experimental measurements for a set of parameters of interest. The idea can also be combined with different architectures of convolutional neural networks (CNN), recurrent neural networks, and so forth, depending on the application.

To further explore this topic, one can find comparative studies on available operator learning algorithms. Lu et al. [11] compared the relative performance and accuracy of two well-established operator learning methods (DeepONet and Fourier Neural Operator) for different benchmarks. Rashid, Chakraborty, and Anoop Krishnan [12] made a comparison between different neural operator architectures to predict strain in a data-efficient manner in 2D digital composites. Koric et al. [13] applied the DeepONet formulation to solve the stress distribution in homogeneous elastoplastic solids with variable loads and material properties. See also [14–16] for the application of the DeepOnet method and its extension to problems related to solid mechanics.

Despite the benefits of the reviewed work on operator learning, there are some potential drawbacks or new challenges that need to be investigated. First, one still has to deal with the cost of the training which is mainly dedicated to gather valuable data. Second, their rather poor performance once we go beyond the range of training data and third, be sure that any predictions of these algorithms are physically consistent or not. Therefore, researchers introduce physical constraints and expect to reduce the amount of data required to train the deep learning models (see e.g. [17]).

1.2 | Physics-Driven Operator Learning

To address the mentioned issues, another mainstream goes in the direction of integrating the model equations into the loss function of the neural network [18–20]. Considering the concept of Physics-Informed Neural Networks (PINNs) as introduced by Raissi, Perdikaris, and Karniadakis [20], the accuracy of predictions or network results can be significantly improved [21]. Additionally, in scenarios where the underlying physics of the problem is completely known and comprehensive, it becomes possible to train the neural network without any initial data. Among many contributions, for solid mechanics see [21–24], and for constitutive material behavior see [25, 26]. The PINN framework is also extended by using the energy form of the problem known as deep energy method (DEM) [27–30] mixed formulations [21, 31], domain decompositions [32, 33], and other techniques that enhance their predictability [34–36]. In the above-mentioned methods, automatic differentiation is primarily utilized to construct the loss function based on available physical knowledge. The advantage is the so-called mesh-free approach, which can handle complex geometries. The drawback is that the process of automatic differentiation is costly compared with algebraic methods. Furthermore, one often has to adapt the location of collocation points to capture sharp transitions in the solution. Finally, the two mentioned problems with the standard FEM method (i.e., computational cost and being limited to one BVP) still hold for new solvers such as PINN, DEM, and mixed-PINN. In fact, they still cannot compete with FEM for forward problems, although they have the potential to excel in inverse problems where parameter identification is the primary concern.

Therefore, an interesting trend and scientific question is how to combine ideas from operator learning and PINNs. Li et al. [37] combined training data and physical constraints in the context of Fourier neural operators to approximate the solution operator for some popular PDE families. Wang, Wang, and Perdikaris [38] proposed physics-informed DeepONets, which ensure the physical consistency of DeepONet models by leveraging automatic differentiation to impose physical laws via soft penalty constraints. The authors reported a significant improvement in the predictive accuracy of DeepONet models, and they also managed to reduce the need for large training datasets. See also [39–41] for utilizing CNN combined with physical constraints. In a study

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

by Zhang and Gu [42], a PINN tailored for analyzing digitalized materials was introduced, trained without labeled ground data using minimum energy criteria. For increased efficiency and potentially greater accuracy, Kontolati et al. [43] suggested mapping high-dimensional datasets to a low-dimensional latent space of salient features using suitable autoencoders and pre-trained decoders. Zhang and Garikipati [44] presented encoder-decoder architectures to learn how to solve differential equations in weak form, capable of generalizing across domains, boundary conditions, and coefficients.

While physics-informed operator learning shows promise, efficiently computing derivatives for input variables through automatic differentiation, especially for higher-order partial differential equations, remains a challenge. Furthermore, the presence of discontinuities in the solution makes the training of standard PINN models more challenging. A recent trend seeks to enhance efficiency by approximating derivatives using classical numerical methods, transforming the loss function from a differential equation to an algebraic one. However, this approach may introduce discretization errors, impacting the network's accuracy. Note that the process of training the network using the backpropagation algorithm still relies on automatic differentiation. Research conducted by Ren et al. [45], Zhao et al. [46], and Zhang et al. [47] investigated the potential of physics-driven CNNs utilizing finite difference and volume methods. Fuhg et al. [48] utilized CNN filters adjusted to resemble finite difference operators, performing minimization of energy functionals, which lowers the order of the differential operators compared with residual-based methods. He et al. [49] employed a graph convolutional network in the DEM model to handle irregular domains. Authors also showed that computation of the spatial gradient based on shape functions are more robust and delivers an accurate solution even at severe deformations.

A summary of available solvers and surrogate models, along with their advantages and shortcomings is provided below to clarify our contributions. Standard solvers like finite element and finite difference methods, as well as approaches such as standard PINNs, DEMs, and mixed PINNs, are effective for solving specific PDEs in engineering applications. However, they are limited to single-use solutions for particular problems.

Neural operators aim to address this challenge by learning equations in a parametric way, often relying on heavy input data to effectively learn patterns. To circumvent the costly data generation step for such surrogate models, attention has turned to physics-informed neural operators to solve and provide solutions for a class of problems in a parametric manner. In this context, the term "solver" refers to algorithms capable of finding correct solutions without relying on data from external sources. However, this aspect, despite its importance, is still under heavy development, as training deep learning models efficiently without available data and making them flexible enough to handle sharp transitions in solution fields and complex geometries remains a challenge in many engineering applications.

To address such problems, we propose a new approach heavily motivated by reviewed works. First, we suggest using the discretized formulation from FEM to construct the loss function. As a result, we can handle Neumann boundary conditions and complex geometries much more easier, while also eliminating the need for automatic differentiation in constructing the loss function, thereby increasing the efficiency of the training process. In this method, the physical degrees of freedom are listed as neurons in the output layer. After the training is done, the solution at any arbitrary point within the physical domain can be interpolated using the shape functions of the FEM. In that sense, the method is similar to other neural operators such as DeepOnet, where infinite-dimensional mapping is achieved. Furthermore, we discuss the potential of Fourier-based deep learning models, allowing for the straightforward handling of much more complex topologies in 2D and 3D settings.

The proposed approach has the potential to be directly coupled with many available finite element-based solvers (or other solvers based on finite difference and spectral methods) in a way that it can directly use their residuals as the loss function, avoiding additional implementation efforts. We apply these techniques, named as finite operator learning (FOL), to investigate problems in micromechanics, particularly in the context of heterogeneous microstructures. Our approach maps parametric input field(s), namely elasticity parameters, to solution field(s), that is, deformation components (see Figure 1).

In Section 2, we review the problem formulation, which mainly deals with standard linear mechanics in materials having a heterogeneous microstructure. In Section 3, we focus on the application of deep learning to solve the discretized weak form in a parametric fashion, where the parameter under focus is the distribution of stiffness (i.e., Young modulus) within the microstructure. In Section 4, we present the results as well as additional insights on how to increase the performance and efficiency of the introduced method through the Fourier-based parameterization version. Finally, the paper concludes with discussions on potential future developments.

2 | Problem Formulation

For the mechanical problem in a 2D heterogeneous solid where the position of material points is denoted by $\boldsymbol{X}^T = [x, y]$. By denoting the displacement components by U and V in the x and y directions, respectively. Utilizing the standard FEM, for each element the deformation field, stress tensor $\hat{\boldsymbol{\sigma}}$ as well as elasticity field E, ν are approximated as

$$U = \sum N_i u_i = \boldsymbol{N}\boldsymbol{U}_e \tag{1}$$

$$V = \sum N_i V_i = \boldsymbol{N}\boldsymbol{V}_e \tag{2}$$

$$E = \sum N_i E_i = \boldsymbol{N}\boldsymbol{E}_e \tag{3}$$

$$\nu = \sum N_i \nu_i = \boldsymbol{N}\nu_e \tag{4}$$

$$\hat{\boldsymbol{\sigma}} = \boldsymbol{C}\boldsymbol{B}\boldsymbol{U}_e \tag{5}$$

Here, $\boldsymbol{U}_e^T = \begin{bmatrix} u_1 & \cdots & u_4 \end{bmatrix}$ and $\boldsymbol{E}_e^T = \begin{bmatrix} E_1 & \cdots & E_4 \end{bmatrix}$ are the nodal values of the deformation field in the x direction and elasticity of element e, respectively. Same holds for $\boldsymbol{V}_e$ and ν_e. Moreover,

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

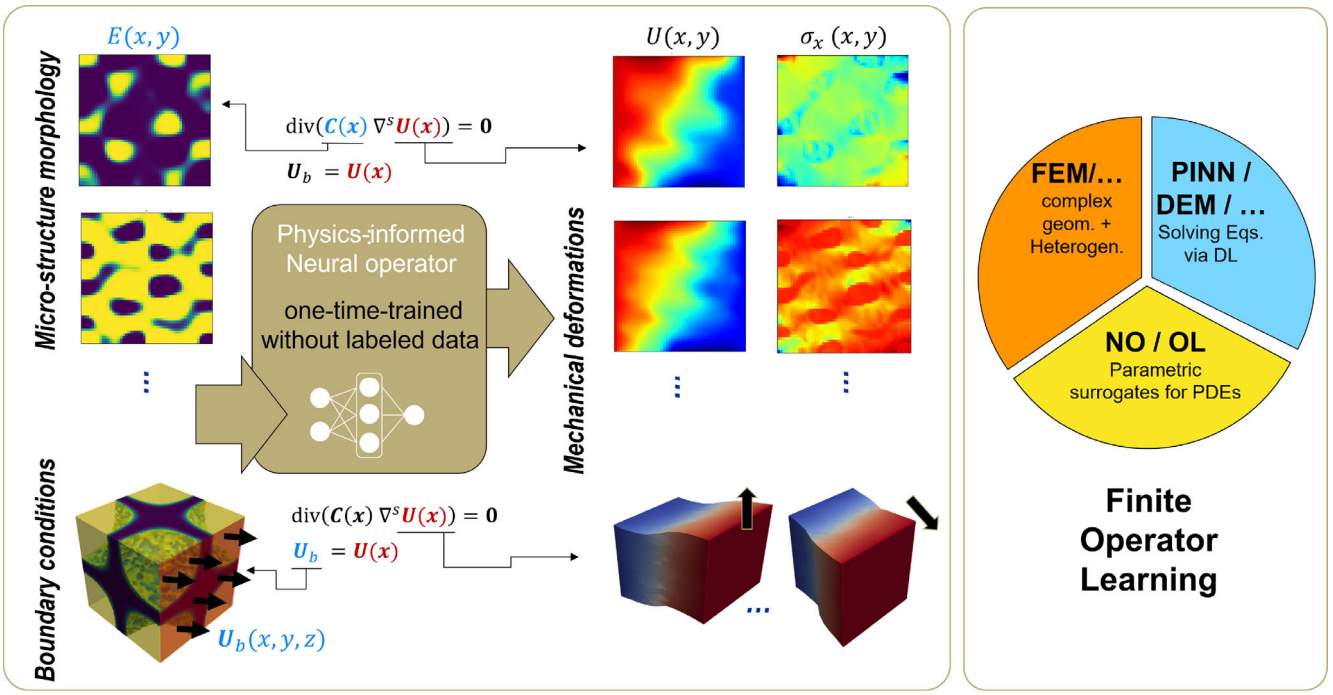


FIGURE 1 | Left: By employing physics-informed operator learning, we aim to map the microstructure property to its mechanical deformation field under specified boundary conditions. Right: the current method, coined as FOL, takes advantage of ideas from the FEM, PINN, DEM, and Neural Operators (NO). In particular cases, the FOL can represent each method individually.

matrices $\boldsymbol{N}$ and $\boldsymbol{B}$ store the corresponding to bi-linear shape functions and their spatial derivatives for a quadrilateral 2D element. Following the standard procedure in the FEM, one can write the so-called *discretized* version of the weak form for one element as

$$\boldsymbol{r}_e = -\int_{\Omega_e} [\boldsymbol{B}]^T \boldsymbol{C}\, [\boldsymbol{B}]\boldsymbol{U}_e\, dV + \int_{\Omega_t} [\boldsymbol{N}]^T \underbrace{\boldsymbol{C}\left[\boldsymbol{B}\boldsymbol{u}_e\right]^T \boldsymbol{n}}_{\bar{t}}\, dS \qquad (6)$$

Here $\bar{t}$ denotes the external traction on Neumann BCs and $\boldsymbol{r}_e$ represents the elemental residual vector which in this case is a 8×1 vector.

In the current work, we are considering a combination of Neumann and Dirichlet boundary conditions, while periodic boundary conditions have not yet been addressed.

3 | A Deep Learning Model to Learn the Static Problem on an Elastic Heterogeneous Domain

3.1 | Neural Network Model

This study relies solely on standard feed-forward neural networks where for each neural network we have the conventional architecture consisting of an input layer, possibly several hidden layers, and an output layer. All layers are interlinked, transmitting information to the subsequent layer. The computation of each component of the vector $\boldsymbol{z}^l$ is expressed as follows:

$$z_m^l = a\left(\sum_{n=1}^{N_l} w_{mn}^l z_n^{l-1} + b_m^l\right), \quad l = 1, \ldots, L \qquad (7)$$

The component w_{mn} shows the weight between the nth neuron of the layer $l-1$ and the mth neuron of the layer l. Every neuron in the lth hidden layer owns a bias variable b_m^l. The number N_I corresponds to the number of neurons in the lth hidden layer. The total number of hidden layers is L. The letter a stands for the activation function.

In a standard PINN framework [20], the network's input is the locations of the collocation points. In the current work of FOL [50], we utilize the so-called *collocation fields*, which constitute randomly generated and admissible parametric spaces used to train the neural network. Here, collocation fields represent possible choices for the elastic properties, that is, $E(x, y)$ and $\nu(x, y)$. More details on how to generate samples are provided in Sections 3.3 and 4.5. Therefore, the input layer consists of information on Young modulus $\{E_j\} = \{E_1, \cdots, E_N\}$ and Poisson coefficient values $\{\nu_j\} = \{\nu_1, \cdots, \nu_N\}$ at all the nodes, and the output layer consists of the components of the mechanical deformation at each discretization node i, that is, $\{U_i\} = \{U_1, \cdots, U_N\}$ and $\{V_i\} = \{V_1, \cdots, V_N\}$. The neural network model is therefore summarized in the following steps which are also depicted on the right-hand side of Figure 2

$$\boldsymbol{X} = \left[\{E_j\}, \{\nu_j\}\right], \quad \boldsymbol{Y} = \left[\{U_i\}, \{V_i\}\right], \quad i, j = 1 \cdots N \qquad (8)$$

$$U_i = \mathcal{N}_{U_i}(\boldsymbol{X}; \boldsymbol{\theta}_{U_i}), \quad V_i = \mathcal{N}_{V_i}(\boldsymbol{X}; \boldsymbol{\theta}_{V_i}), \quad \boldsymbol{\theta}_i = \{\boldsymbol{W}_i, \boldsymbol{b}_i\} \qquad (9)$$

Here, the trainable parameters of the ith network (i.e., $\boldsymbol{W}_i$ and $\boldsymbol{b}_i$) are denoted by $\boldsymbol{\theta}_i$. Note that we propose employing separate fully connected feed-forward neural networks for each output variable. The number of nodes is denoted by N, set to 121 in this paper.

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

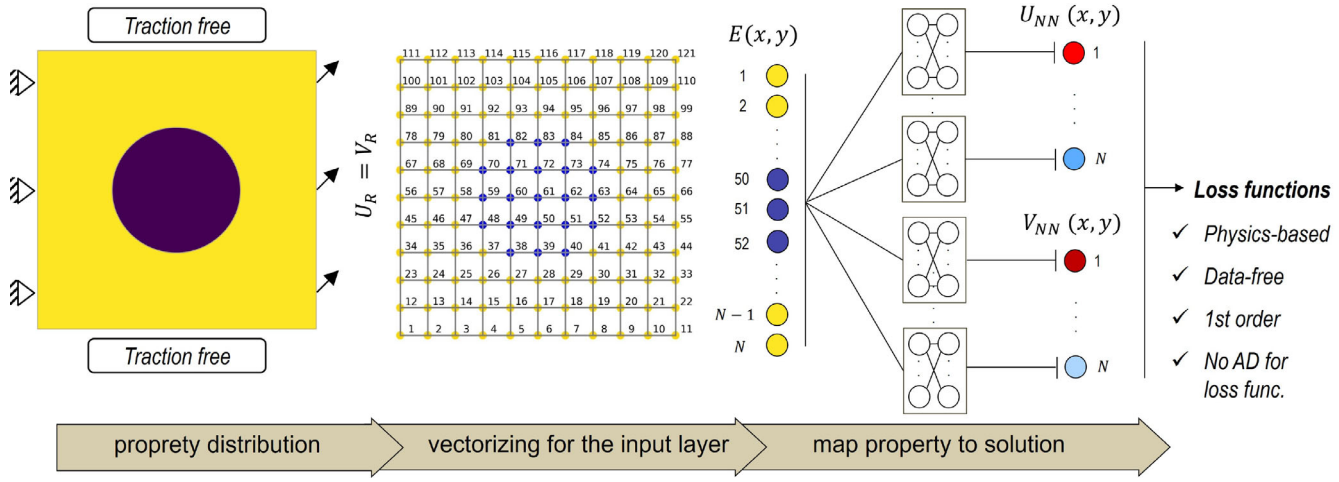


FIGURE 2 | Network architecture for finite operator learning (FOL), where information about the Young modulus distribution goes in and the unknown displacement components are evaluated utilizing a series of separate deep neural networks. For constructing the physical loss function, automatic differentiation (AD) is not required.

The domain remains square-shaped. However, the formulation is very general and can be easily adapted for irregular geometries and unstructured meshes, thanks to the power of finite element techniques [51].

Note that in Figure 2 and throughout the rest of the work, we assume that the Poisson ratio remains constant in the RVE, and our focus lies solely on the variation of Young's modulus. The latter assumption helps to simplify the input parameter space but does not constrain the methodology. Next, we introduce the loss function. We assume traction-free boundary conditions on the top and bottom edges, resulting in vanishing traction at the Neumann boundaries. The displacement components at the left edge are fixed (i.e., $U_L = V_L = 0$). On the right edge, we apply tension and shear simultaneously (i.e., $U_R = V_R = 0.05$ mm on Figure 2) leading to a rather general Dirichlet boundary condition.

The total loss term $\mathcal{L}_{tot}$ combines both the elemental energy form of the governing equation $\mathcal{L}_{en,e}$ and the Dirichlet boundary terms $\mathcal{L}_{db,i}$. It is important to note that, thanks to the weak formulation, the Neumann boundary conditions are automatically included in $\mathcal{L}_{en,e}$. Following the approach outlined in [50] for thermal problems, the total loss function involves the integration of element residual vectors (i.e., Equation 6) using Gaussian integration, resulting in

$$\mathcal{L}_{tot} = \sum_{e=1}^{n_{el}} \mathcal{L}_{en,e}(\boldsymbol{\theta}) + \sum_{i=1}^{n_{db}} \mathcal{L}_{db,i}(\boldsymbol{\theta}) \tag{10}$$

$$\mathcal{L}_{en,e} = \boldsymbol{U}_e^T(\boldsymbol{\theta})\left[\boldsymbol{K}_e\, \boldsymbol{U}_e(\boldsymbol{\theta})\right] \tag{11}$$

$$\mathcal{L}_{db,i} = \frac{1}{n_{db}}\left|U_i(\boldsymbol{\theta}) - U_{i,db}\right| + \frac{1}{n_{db}}\left|V_i(\boldsymbol{\theta}) - V_{i,db}\right| \tag{12}$$

$$\boldsymbol{K}_e = \sum_{n=1}^{n_{int}} \frac{w_n}{2}\, \det(\boldsymbol{J})\, \left[\boldsymbol{B}(\xi_n)\right]^T \boldsymbol{C}_e(\xi_n)\boldsymbol{B}(\xi_n) \tag{13}$$

$$\boldsymbol{C}_e(\xi_n) = \frac{\boldsymbol{N}(\xi_n)\boldsymbol{E}_e}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \tag{14}$$

Here, $n_{int} = 4$ represents the number of Gaussian integration points. Additionally, ξ_n and w_n denote the coordinates and weighting of the nth integration point and the determinant of the Jacobian matrix is denoted by $\det(\boldsymbol{J})$. In Equation (12), $n_{db} = 22$ denotes the number of nodes along the Dirichlet boundaries (i.e., the left and right edges). Additionally, $U_{i,db}$ and $V_{i,db}$ represent the given displacement at node i. The final loss function becomes

$$\mathcal{L}_{tot} = a_e \sum_{e=1}^{n_{el}} \boldsymbol{U}_e^T \left[\sum_{n=1}^{n_{int}} \boldsymbol{B}^T \boldsymbol{C}_e \boldsymbol{B}\right] \boldsymbol{U}_e + a_b \sum_{i=1}^{n_{db}} \left|U_i - U_{i,db}\right| + \left|V_i - V_{i,db}\right| \tag{15}$$

In the above relation, the determinant of the Jacobian matrix is constant since we use quadrilateral elements. Additionally, for the four integration points, we have $w_n = 1$. Based on the energy formulation for the first term in Equation (15), one can write $a_e = \frac{w_n}{2}\det(\boldsymbol{J})$. For the second term, we suggest $a_b = \frac{A}{n_{db}}$, which needs to be adapted based on the problem via a parameter study or novel adaptive weighting techniques. Note that this term can be limited by applying a hard boundary constraint. In Equation (15), a_b represents the weighting of the boundary terms which guarantees the enforcement of the boundary conditions. Note that in Equation (15), we utilized the discretized energy form, but one can also use the norm of the total residual vector. The total loss term $\mathcal{L}_{tot}$ is minimized with respect to the network parameters $\boldsymbol{\theta}$ concerning a batch of collocation fields.

3.2 | Short Review on the DeepONet Model

Building on the foundational principles of the universal approximation theorem [52], the so-called vanilla DeepONet architecture [8] with two outputs is used here. The network architecture consists of a single branch and single trunk network as depicted in Figure 3.

To obtain multiple outputs from a single DeepONet, we divide the final layers of the branch and trunk networks into two parts of equal length (see also [53]). Two operators that map the property distribution to the mechanical deformations are written as

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

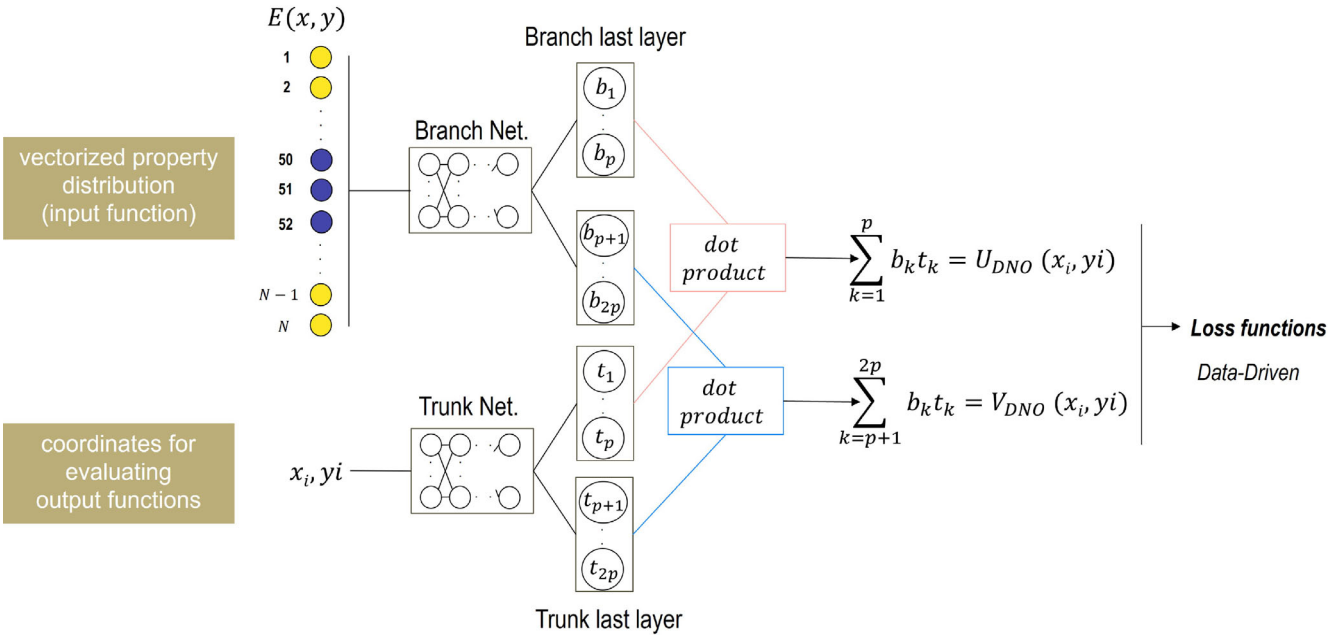


FIGURE 3 | DeepONet's architecture for translating property maps into mechanical deformations. The property map is vectorized and sampled at nodes (acting as sensors) to serve as input to the branch network, while the trunk network receives coordinates for output evaluation. Both networks may have different hidden layers, but end up with the same number of neurons in their final layers, which are divided into two equal-length vectors. The dot product of these vectors produces the desired outputs (U_i and V_i).

$$\boldsymbol{G}_\theta^{(U)}(\boldsymbol{E})(\boldsymbol{X}) = \sum_{k=1}^{p} b_k\left(E\left(x_1 y_1\right), E\left(x_2, y_2\right), \ldots, E\left(x_N, y_N\right)\right) \cdot t_k(\boldsymbol{X})$$
$$\boldsymbol{G}_\theta^{(V)}(\boldsymbol{E})(\boldsymbol{X}) = \sum_{k=p+1}^{2p} b_k\left(E\left(x_1 y_1\right), E\left(x_2, y_2\right), \ldots, E\left(x_N, y_N\right)\right) \cdot t_k(\boldsymbol{X}) \tag{16}$$

In Equation (16), $\boldsymbol{G}^{(U)}$ and $\boldsymbol{G}^{(V)}$ denote the solution operators for displacements in the x and y directions, respectively. Finally, for the pure data-driven DeepONet, the total loss function is written as

$$\mathcal{L}_{DeepONet} = \frac{1}{I_{train}\, N} \sum_{i=1}^{I_{train}} \sum_{j=1}^{N} \left| \boldsymbol{G}_\theta^{(U)}\left(\boldsymbol{E}^{(i)}\right)\left(\boldsymbol{X}_j^{(i)}\right) - U_j^{(i)} \right|^2 + \left| \boldsymbol{G}_\theta^{(V)}\left(\boldsymbol{E}^{(i)}\right)\left(\boldsymbol{X}_j^{(i)}\right) - V_j^{(i)} \right|^2 \tag{17}$$

For more details about the DeepONet training see Appendix A. It is worth noting that several enhancements have been introduced to improve the performance of the original DeepONet architecture [53, 54].

3.3 | Preparation of Collocation Fields

The neural network mentioned above should be trained using initial input samples representing the Young modulus distribution. It is important to note that the training process is entirely unsupervised as no FEM solution is required and random fields are sufficient to initiate the training by having the loss function based on discretized FEM residual vector.

In the current work, we consider two-phase heterogeneous materials, which is a common example in many alloys and composite materials [55, 56]. The goal is to modify the microstructure's morphology and properties, allowing for arbitrary shapes and volume fraction ratios between the two phases. However, the elastic properties (i.e., Young's modulus and Poisson ratio) of each phase remain fixed. Additionally, to reduce complexity, we shall investigate at first only one ratio between the elastic properties of each phase.

So far, we have the following model simplifications: keeping the Poisson ratio and phase contrast ratio constant, and using a grid of 11 by 11. Even with the previously mentioned simplifications, there can be approximately $2^{121} \approx 2.6 \times 10^{36}$ independent samples. Creating all these samples is unfeasible and may be unnecessary depending on considered application. Hence, the user should determine the relevant parameter space, specific to each problem, thereby preventing unnecessary complications. The latter point can be a challenging aspect in designing a very general deep learning model that works for all cases.

As mentioned, our preliminary motivation relies on two-phase metallic materials, and after some investigations, we have created simply a rather meaningful set of samples for training [50]. We generated 4000 samples for a grid of size $\sqrt{N}$ by $\sqrt{N}$, with $\sqrt{N} = 11$. On Figure 4, where the generated microstructures are sorted based on their phase fraction value, six samples with low and high phase fraction values are shown.

To assess the NN's performance, additional samples are chosen for testing, five of them are shown in Figure 5. These samples are entirely novel and deliberately selected for their symmetrical aspects, ensuring they are beyond the scope of the training dataset (see also Figure B2 in Appendix B for an overview on the topology of the samples). These samples are motivated by dual-phase

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

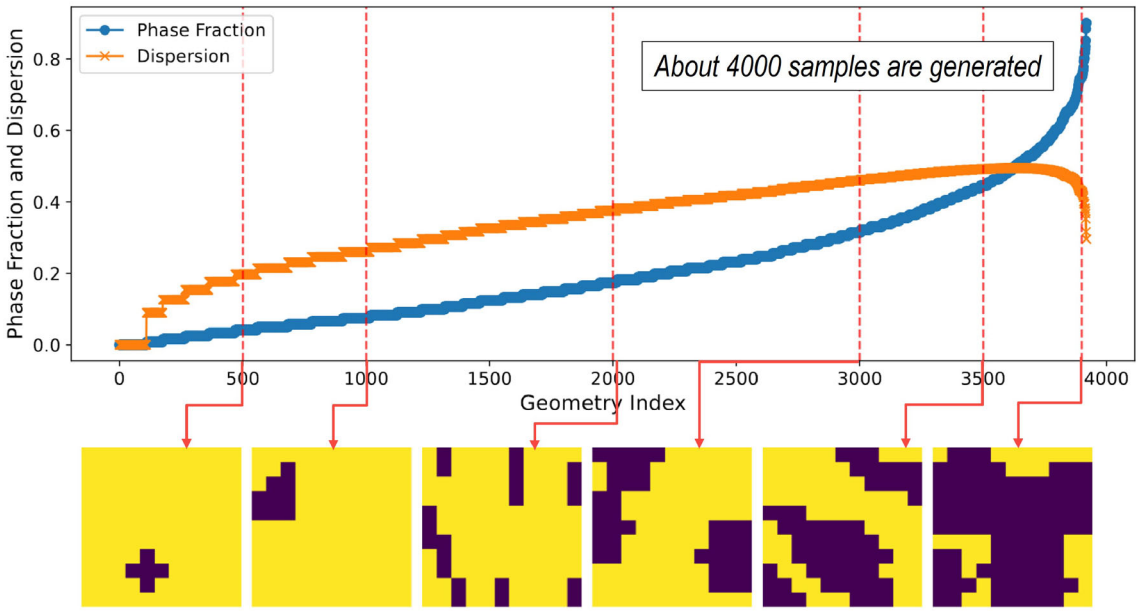


FIGURE 4 | Examples showcasing created morphologies for a two-phase microstructure.

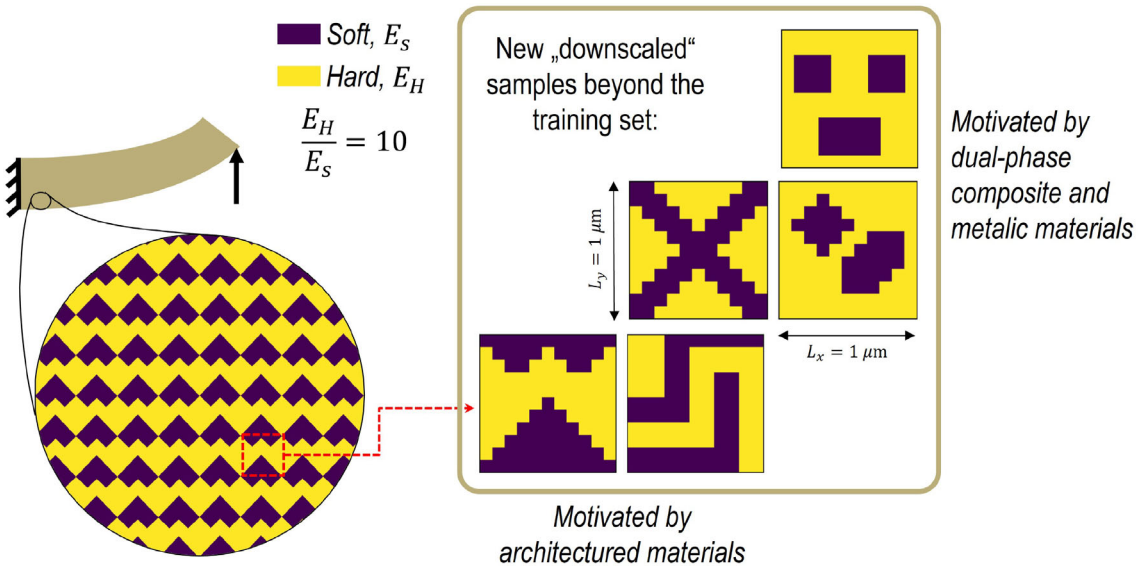


FIGURE 5 | Unseen samples used for testing the neural network after training.

architectures in composite materials, where a soft and hard phase are combined. Note that all the samples are downsized to be represented by an 11 by 11 grid. The downsampling strategy is also described in [50], where a simple CNN algorithm combined with a max-pooling is used [57].

4 | Results

The algorithms developed in this study are implemented using JAX [58] software. The codes are also implemented in SciANN package [59], and the methodology can be adapted to other programming platforms as well. A summary of the network's (hyper)parameters is presented in Table 1. Note that whenever multiple options for a parameter are introduced, we examine the influence of each parameter on the results. Training can also be done on personal laptops without relying on GPU acceleration, thanks to the efficiency of the proposed framework. Nevertheless, we primarily used a Quadro RTX 6000 GPU unless otherwise noted.

The material parameters listed in Table 2 do not necessarily represent a specific material at this point. For the chosen Young modulus values and boundary conditions we did not see the necessity to perform any additional normalization before the training. In all the results provided below, the units for displacement-based quantities are in micrometers (μm), while for stress-based quantities, they are in gigapascals (GPa).

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 1 | Summary of the network parameters.

Parameter	Value
Inputs, outputs	$\{E_j\} = \{E_1, \cdots, E_N\}$, $[\{U_i\} = \{U_1, \cdots, U_N\}$, $\{V_i\} = \{V_1, \cdots, V_N\}]$
Activation function	Swish [60]
Layers and neurons for each sub-network	$(L, N_l) = (2, 10)$
Optimizer	Adam [61]
(Number of samples, batch size)	(100,2), (1000, 25), (2000, 50), (4000, 100)
(Learning rate α, number of epochs)	$(5 \times 10^{-4}, 4000)$

TABLE 2 | Model input parameters for the mechanical problem.

	Young modulus value/unit
Phase 1 (E_H)	1.0 GPa
Phase 2 (E_S)	0.1 GPa
Applied displacement, small defo. model (U_R, V_R)	(0.05 μm, 0.05 μm)

4.1 | Evolution of Loss Function and Network Prediction

The evolution of each loss term is illustrated in Figure 6 throughout epochs. On the left-hand side, the loss terms associated with the physics-driven model are presented, while the right-hand side displays the loss evolution for the supervised data-driven model. It is noteworthy that satisfactory results are achieved only after 4000 epochs in both cases. It is important to mention that, for the data-driven model, offline finite element calculations are performed on the same set of 4000 samples, and the identical network architecture is employed as in the physics-driven model.

In Figure 6, we employ the "swish" activation function, and 4000 samples with a batch size of 100 while the remaining parameters align with those in Table 1. All the loss functions decay simultaneously, and we did not observe any significant improvement after 4000 epochs.

For the loss term related to the Dirichlet boundary conditions, we assigned higher weightings to satisfy the boundary terms. Following a systematic study that began with equal weightings for both loss terms, we selected $A = 10\, n_{nb}$, indicating $a_b = 10$. Choosing equal or lower weightings resulted in less accurate predictions within the same number of epochs as the boundary terms were not fully satisfied. Opting for higher values did not significantly improve the results. It is worth mentioning that with the current approach, one can also easily apply hard boundary constraints by omitting nodes (i.e., degrees of freedom) with Dirichlet boundary conditions from the output layers. We intentionally did not perform this for the current study to demonstrate that even with simple soft constraints, one can train the neural network properly. This aspect is crucial for extending the neural network to learn different boundary conditions in a parametric way for future work.

In Figure 7, the predictions of the same neural network are depicted. In the middle part of the figures, we display the reference solutions obtained by the FEM. All the results are assessed on finer meshes (100×100) using bi-linear shape function interpolations. On the right-hand side, the absolute value of the point-wise difference between FOL and FEM is shown. All displacement values are denoted in micrometer (μm), stress components are represented in (GPa), and Young modulus is expressed in (GPa) too.

The predictions of the neural network for unseen microstructures are in very good agreement with those from the FEM. The maximum pointwise error for displacement components remains below 10%, localized only in some points of the entire domain. For the stress components, which are obtained through post-processing of deformations values, the errors are higher, as expected, since the neural network predictions are solely based on deformations, and any higher-order derivatives are more sensitive to slight deviations in the original solution. On the other hand, for many applications in multiscale material modeling, homogenized values of stress and strain components are needed, and these are well-predicted as outlined in Section 4.2.

Based on the reported results, we observe a tendency in the error distribution depending on the material topology. The main critical regions with higher errors are localized close to the phase boundaries where a jump in the solution is expected. A more quantitative report on the errors and possible remedies to decrease them is discussed in the following sections. In particular, we will discuss the influence of initial sample fields, using a pre-trained autoencoder, as well as a Fourier-based parameterization to reduce these errors.

4.2 | The Effect of the Number of Initial Samples

In Figure 8, we not only compare the performance of the data-driven and physics-driven FOL models but also study the influence of the initial training samples.

Note that other (hyper)parameters of the NNs are kept the same for this study (see also Table 1). As expected, reducing the number of samples from the initial dataset leads to increased errors.

What is noteworthy from such observations is how accurately the physics-driven model can predict the solution even when trained based on 2000 initial random samples. This confirms that providing equations for training has a more significant impact on the network performance. To confirm this observation, we tested several cases with different complexities in the next section.

In Figure 9, we present quantitative error measurements by using different error metrics for four distinct unseen microstructure. Error metrics are defined as: $Err_{MSE} = \sqrt{\frac{1}{N}\sum_i \left(u_{NN}(i) - u_{FE}(i)\right)^2}$ or as $Err_{MAX} = Max\left(|u_{NN}(i) - u_{FE}(i)|\right)$. The former indicates the average pointwise error, while the latter focuses on the

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

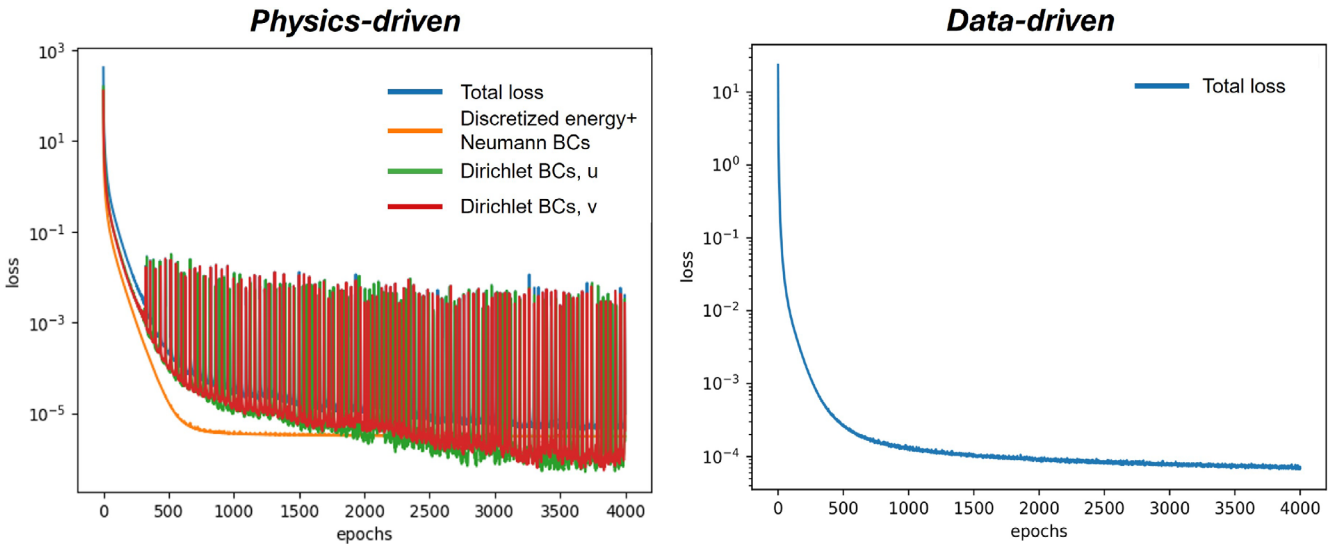


FIGURE 6 | Evolution of loss terms for parametric learning of the quasi-static mechanical equilibrium problem.

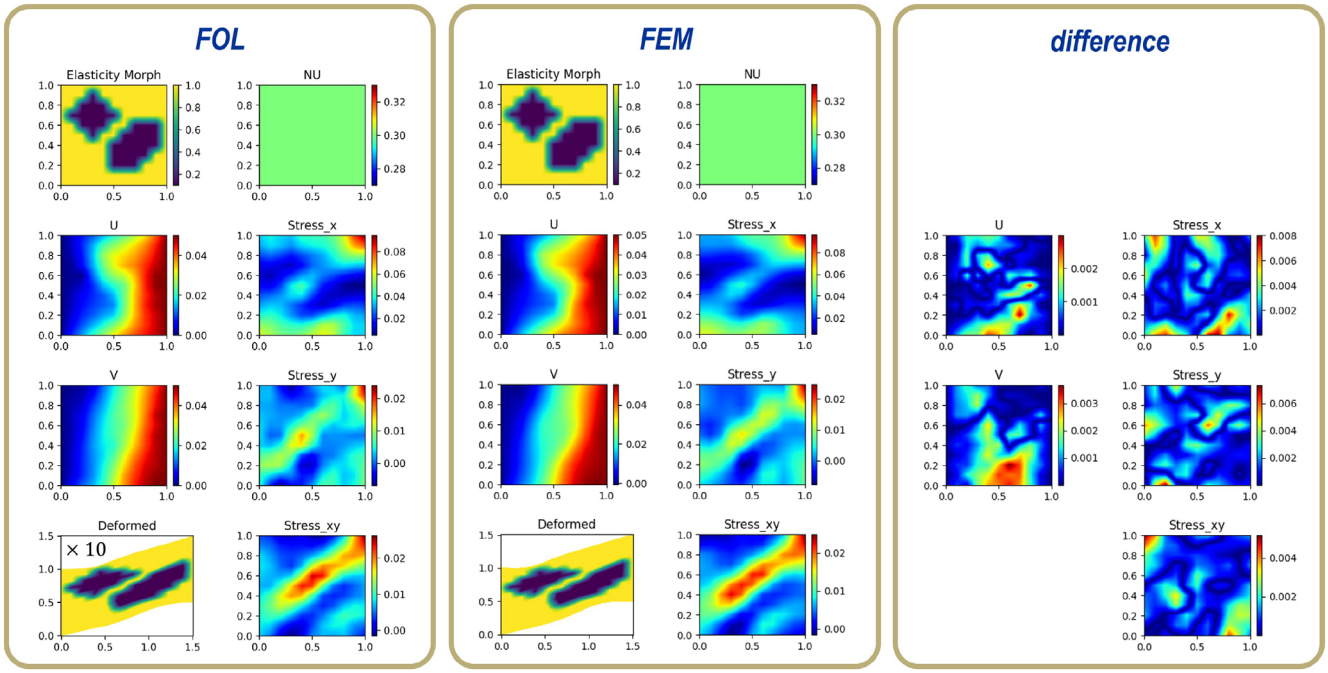


FIGURE 7 | Predictions of the trained FOL for approximately circular and ellipsoidal inclusions.

maximum local difference. Training the model with more samples enhances its performance, and one can observe a convergence behavior similar to that of other numerical methods, despite are based on a different concept here. Note that the selected test cases were intentionally chosen to represent a range of errors: minor, medium, and maximum. For example, Test 1 has a topology similar to the training data, while Tests 3 and 4 involve topologies that are significantly different from the training set, featuring sharper and more complex features.

From the comparison of error plots, two interesting points emerge. First, the errors for the physics-driven model are generally lower than those for the data-driven model, regardless of the number of sample fields used for training the model. Second, for test case 1, which is more similar to the training samples, the errors of the data-driven model are very acceptable. This indicates that, apart from the quantity of the samples, the so-called quality of them plays a role. In other words, even if it sounds rather unfeasible to train the model for any possible distribution of Young modulus, one can always select a meaningful set of samples for each specific application to train the model properly. For test cases 3 and 4, we observe higher errors with both error metrics, as they are significantly distant from the shapes of training samples. Therefore, we expect that by further enriching the initial sample fields, the errors for these two cases will also decrease.

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

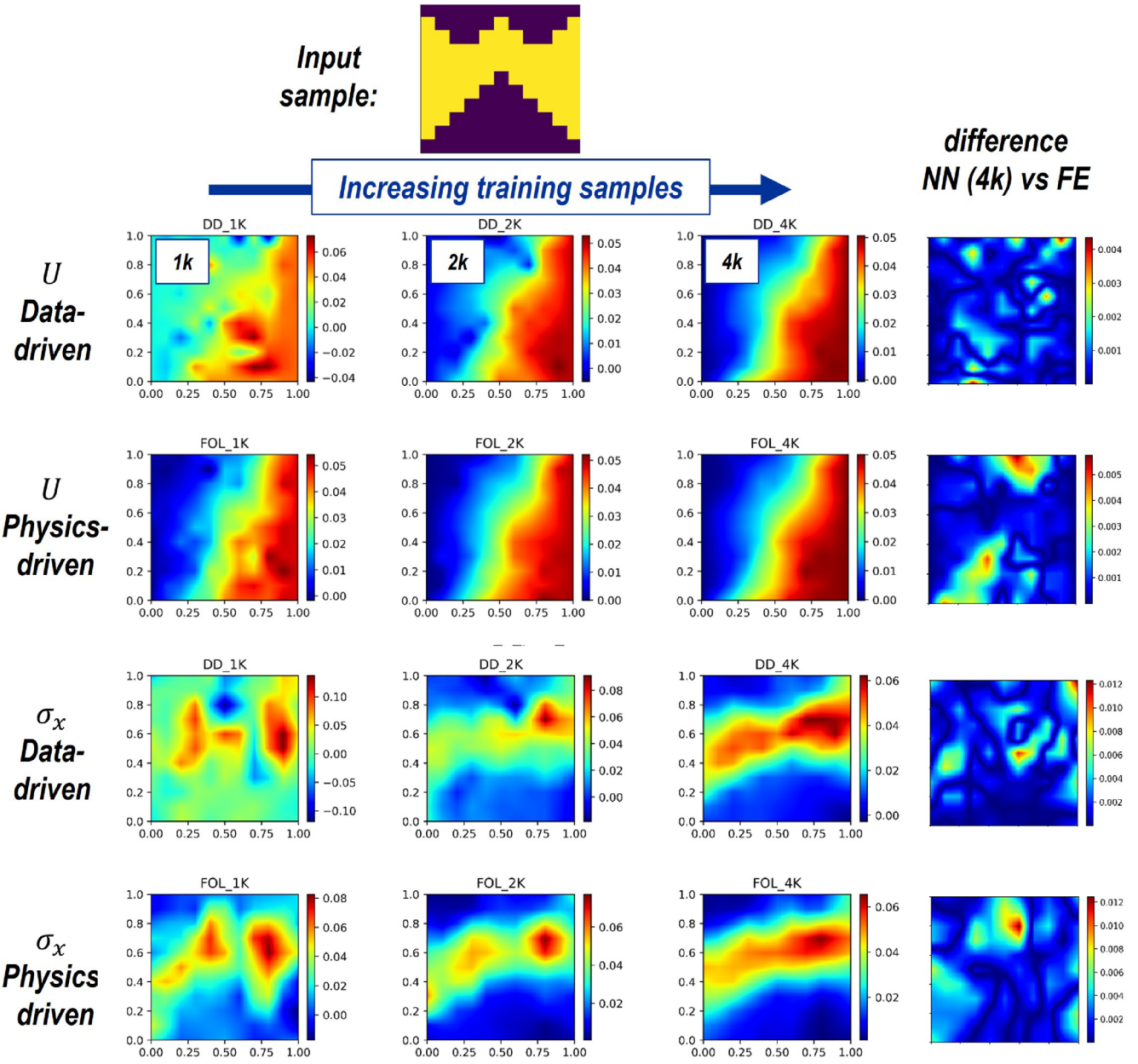


FIGURE 8 | Performance of data-driven and physics-driven approaches for the different number of training samples.

4.3 | Comparison With Other Neural Operator Techniques

In this section, we compare the results of the introduced FOL method against one particular data-driven operator learning method, DeepONet. For a fair comparison, the DeepOnet model explained in Section 3.2 is trained by using 4000 samples, the same data set discussed in the last section. The comparison results, illustrated in Figure 10 for several unseen test cases, reveal that the physics-driven method provides more accurate predictions for mechanical deformation as well as stress distribution. Note that the DeepOnet performance on predicting the displacement values is still acceptable to some extent, as DeepOnet is designed to achieve this goal. The stress values are obtained by taking the derivatives of the displacement field after training and there are no additional loss terms for them. This is one of the main reasons behind the larger deviations in predicting the stress values, as any fluctuations in the displacement predictions heavily influence the stress values.

This highlights the attractiveness of the physics-driven FOL for two main reasons: First, it operates without the need for labeled data during training, and second, it provides more accurate predictions for unseen cases for both displacement components (as primary variables) and their spatial derivatives (i.e., stress components). Nevertheless, it is essential to emphasize that supervised training incurs a lower training cost, averaging approximately 7 to 10 times less compared with the physics-based approach.

The observations discussed above are not limited to a specific test case; they consistently appear across other unseen cases and regions as well. It is noteworthy that in future works, other

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

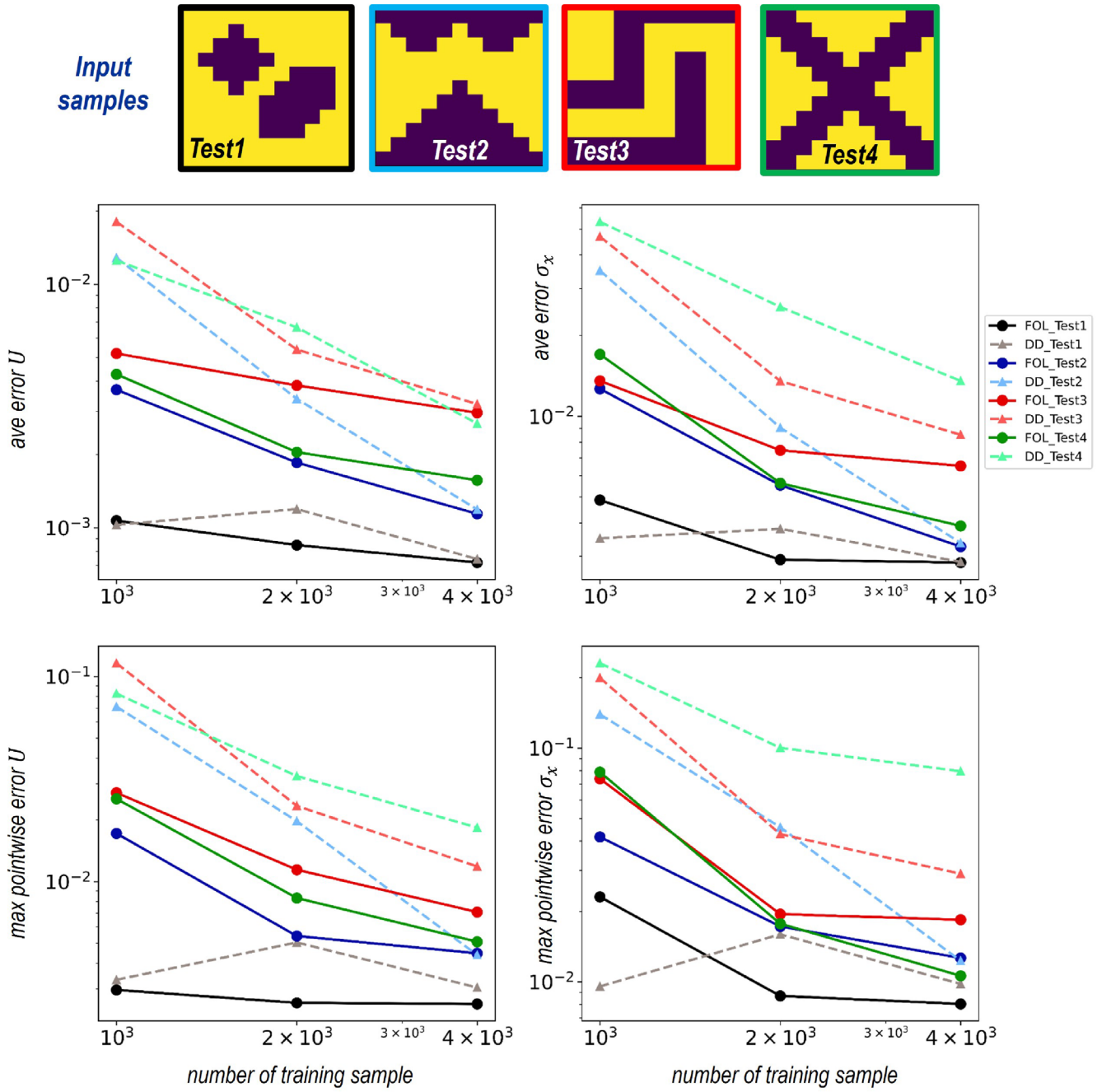


FIGURE 9 | Improving the performance of the deep learning models by increasing the number of randomly generated samples for training. Different error evaluations are utilized based on different applications, where the physics-driven model always behaves better than the data-driven one for the same unseen test cases.

operator learning approaches should be properly compared under the same data/sample set to enable a fair comparison. Moreover, the DeepONet can be trained directly with stress data to predict stress field more accurately [13]. We use the same samples for both FOL and DeepOnet to ensure a fair comparison. However, the performance of data-driven DeepOnet models would also improve if trained on more data. The accuracy of DeepOnet can potentially be enhanced by using recent extensions that leverage addition and subtraction operations [54].

4.4 | Influence of Phase Contrast on the Results

In this section, we briefly discuss the changes in the phase contrast (ratio of the Young modulus) based on the obtained results from the FOL. So far, we mainly focused on the ratio of $E_H/E_S = 10$. In Figure 11, we also showed the averaged error for the displacement component U in the x direction as we increase the stiffness ratio to $E_H/E_S = 20$ or decrease it to $E_H/E_S = 2$. These latter values are roughly representative of some engineering applications in composite and dual-phase metallic materials.

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

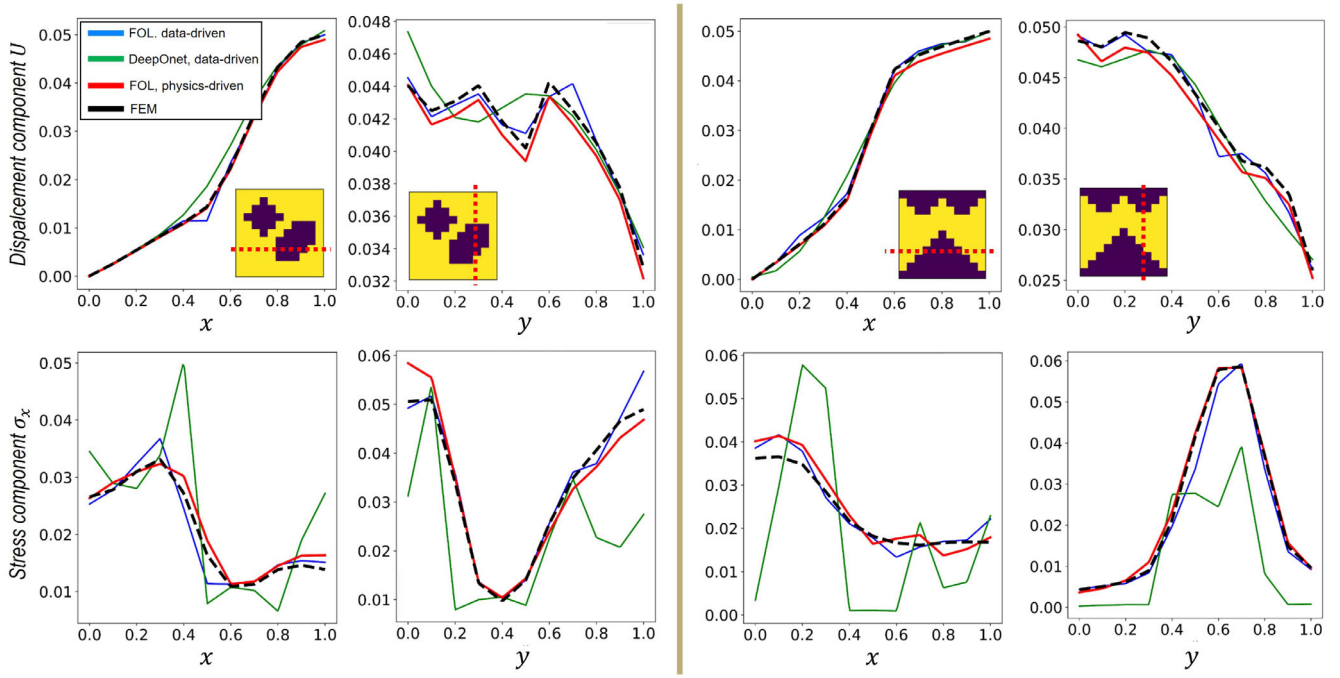


FIGURE 10 | Comparison of data-driven FOL, data-driven DeepOnet, and physics-driven FOL approaches regarding the displacement component in x direction U as well as stress component σ_x along x and y directions, respectively.

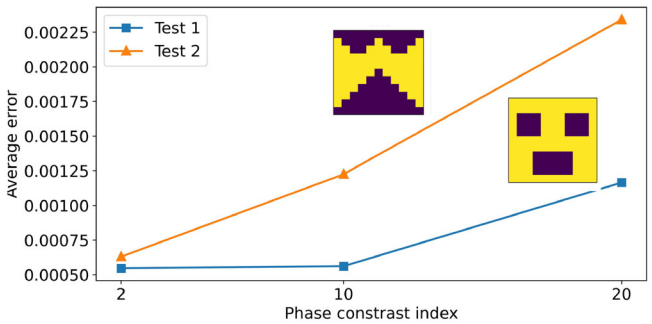


FIGURE 11 | Influence of ratio of phase contrast on the obtained results.

We can conclude that by increasing the ratio, the errors generally tend to slightly increase. This is attributable to the anticipation of higher jumps and greater variations in nonlinearities and complexities in the deformation profiles. Also, note that the morphology of unit cell 2 presents more interfaces between both phases than unit cell 1, which can be one reason for higher errors in this test case. Nevertheless, we emphasize that the obtained results also depend on the shape and topology of the limited random samples chosen for the training. Therefore, one cannot simply conclude any relation between the topology of the microstructure and the accumulated error.

4.5 | Discussions on Up-Scaling Results and Mesh Convergence

To minimize training costs, the microstructure morphologies (i.e., Young modulus distributions) undergo downsampling onto a smaller grid. The question arises regarding potential information loss and methods to reconstruct the original solution. In Figure 12, various approaches are demonstrated for this purpose. The first method involves linear interpolation of the solution in the domain. Quadratic interpolation is also explored, but despite its smoothness, it does not provide significant advantages. Certainly, the most effective yet more costly strategy involves utilizing a pre-trained autoencoder, as shown on the right-hand side of Figure 12.

We report the results of a pre-trained model, referred to as the microstructure-embedded autoencoder (MEA). The original idea and similar works are introduced and reviewed in [62] and is summarized in Figure 13. First, the MEA method condenses a high-resolution parametric space, which represents the spatial distribution of mechanical properties for two-phase composite materials, into low-resolution grids. Second, it employs a pre-trained FOL to solve the BVP on the coarsest grid. Finally, the solution is refined using the MEA architecture to produce a high-fidelity output by concatenation of parametric space fields of various resolutions in the decoder section. The training of such an autodecoder requires solution data of the parameter space in different resolutions. The training of MEA with 5400 data sets takes 30 min on a workstation working with a Quadro RTX 8000 GPU.

Figure 14 shows a significant error reduction compared with the simple interpolation approach using the MEA method. Despite its simplicity, it is worth mentioning that the interpolation approach is both cost-effective and implementable, resulting in an acceptable agreement when only averaged values of upsampled solutions are required for specific applications, such as homogenization. However, the MEA approach can capture the sharp solution transitions, which are important for the

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

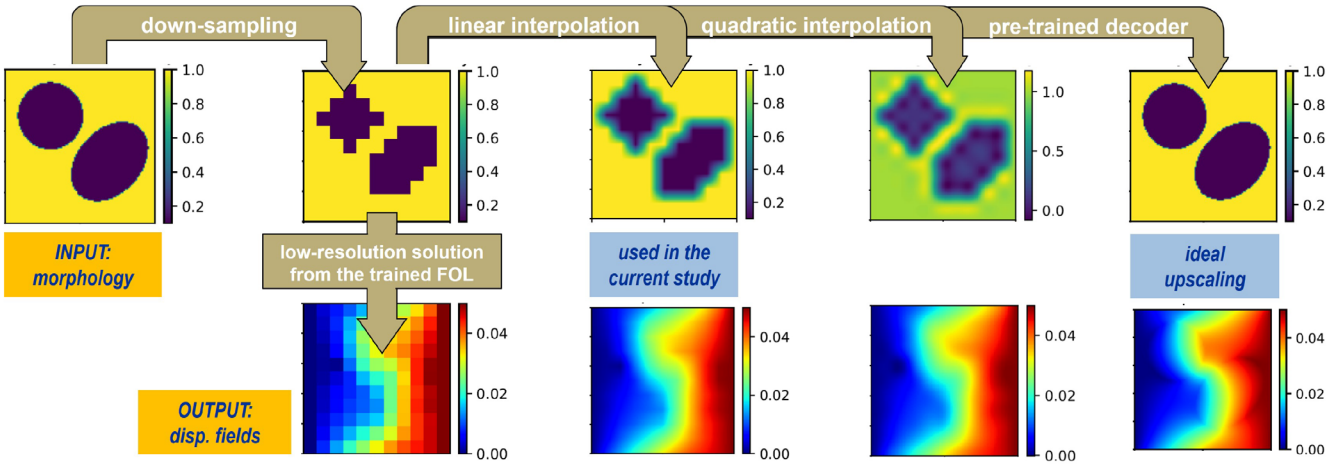


FIGURE 12 | Different strategies for upsampling the results (displacement component U).

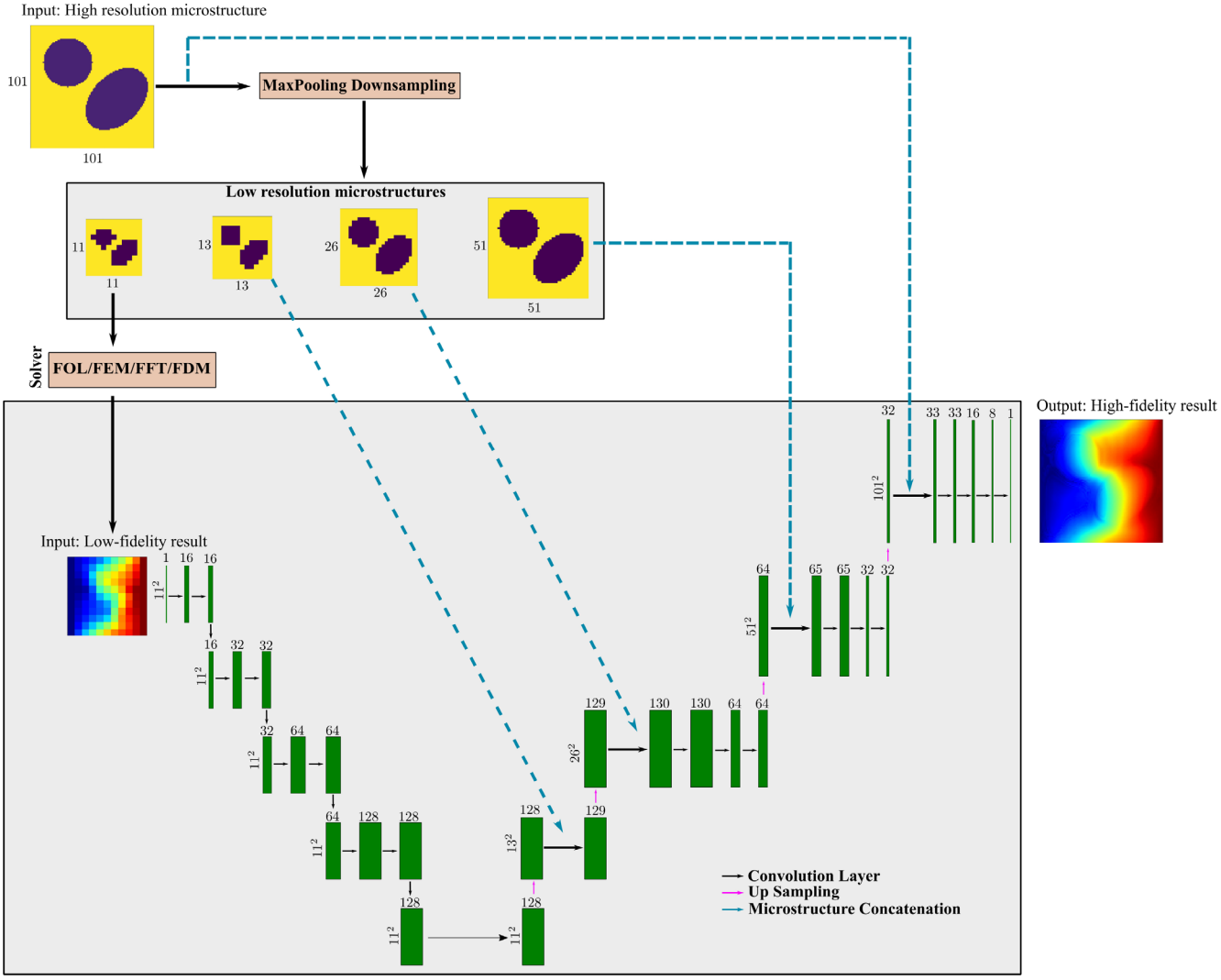


FIGURE 13 | In the developed MEA architecture, the process begins by condensing a high-resolution mechanical property map into various lower-resolution maps. The next step involves using the coarsest grid to solve the boundary value problem. The resultant low-resolution output is upscaled through an enhanced autoencoder.

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

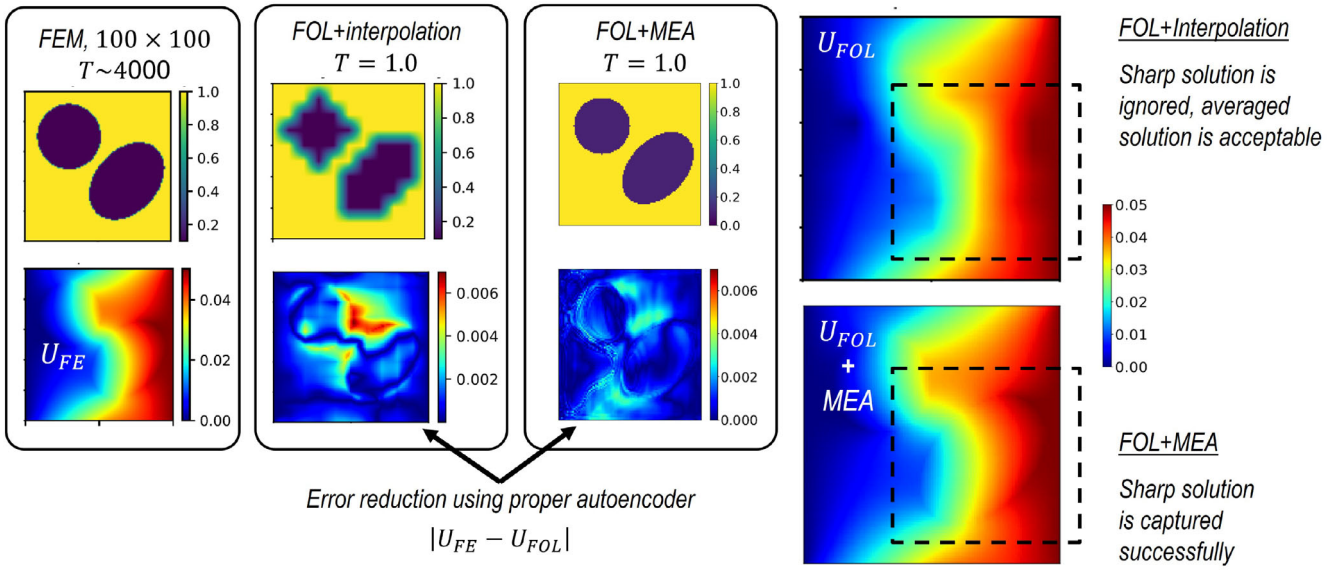


FIGURE 14 | Obtained results for the displacement component U using the MEA approach show that errors are significantly reduced on a finer grid while reducing the computational cost substantially.

evaluation of local stress fields. See also Figure B1 in Appendix B for more test cases using the introduced MEA approach.

As shown above any neural operator techniques can be solely trained on the reduced space and then the information can be reconstructed via an additional deep learning model. Nevertheless, in some applications, it is much easier to train the neural operator based on the reduced set of input features from the beginning to avoid the above two-step process. The latter idea is discussed in the following section.

4.6 | Fourier-Based Parameterization for FOL: Mesh Convergence and Extension to 3D

A possible shortcoming of the original FOL architecture introduced in Figure 2 is its training cost for higher mesh densities, as the number of neurons in the input and output layers increases based on the number of D.O.F. of the model. The latter is also a limiting factor for other neural operator techniques, as all of them require being fed a sufficient number and variety of input samples. Similar to the idea of parametric space reduction by means of simple downsampling strategies discussed earlier in this work, we propose an alternative way to significantly simplify the FOL architecture. As a result, we enhance the capability of the method to handle higher mesh densities and even 3D problems without a significant increase in training time.

This idea is based on the parametric representation of the inputs via a Fourier transformation where a combination of different frequencies for sinusoidal and cosinusoidal functions forms the input layer [63]. As depicted in Figure 15, we can reduce the number of inputs to M desired combinations of frequencies, which is usually a much smaller than the number of degrees of freedom in the system, denoted by N. Moreover, we take advantage of applying hard boundary constraints, and therefore only the first term in Equation (15) is used for minimization. We use $U_R = -0.05$ and $V_R = 0.1$ mm to show the flexibility of the method to handle other BCs. The application of hard boundaries in the FOL method is very straightforward, where we simply omit those degrees of freedom with Dirichlet boundary terms from the system of equations. This approach and similar ones are based on ideas from the implementation of the FEM. In other words, we do not need any extra modification of the values in the output layer, similar to other studies on the application of hard boundaries in the PINN framework (see [64] and references therein).

The Fourier-based function for representation of the elasticity map in a 2D setting has the following form through a combination of sine and cosine functions:

$$E_f(x,y) = \sum_{i}^{n_{sum}} \left[c_i + A_i \sin\left(f_{x,i}\, x\right)\cos\left(f_{y,i}\, y\right) + B_i \cos\left(f_{x,i}\, x\right)\sin\left(f_{y,i}\, y\right) + C_i \sin\left(f_{x,i}\, x\right)\sin\left(f_{y,i}\, y\right) + D_i \cos\left(f_{x,i}\, x\right)\cos\left(f_{y,i}\, y\right)\right] \quad (18)$$

In Equation (18), $\boldsymbol{x} = (x,y)^T$, c_i is the real-valued constant, $\{A_i, B_i, C_i, D_i\}$ represents the amplitudes of the corresponding frequency number and $\{f_{x,i}, f_{y,i}\}$ represents the frequency in the x and y-direction. Finally, we specify the property function E_f through a so-called Sigmoidal projection as shown below to have more realistic and interesting characteristics for the microstructure.

$$E(x,y) = \left(E_{max} - E_{min}\right)\cdot \mathrm{Sigmoid}\left(\beta\left(E_f - 0.5\right)\right) + E_{min} \quad (19)$$

With the above projection, we ensure that the value of Young's modulus lies between $E_{min} = 0.1$ and $E_{max} = 1.0$ MPa. With the β parameter, one can also control the transition between the two phases, which is set to $\beta = 1$ for the samples in this study. Some of the main modes, as well as possible samples that can be processed, are shown in the lower part of Figure 15.

The new design for the Fourier-based parameterization for FOL framework can therefore be summarized below:

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek, 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

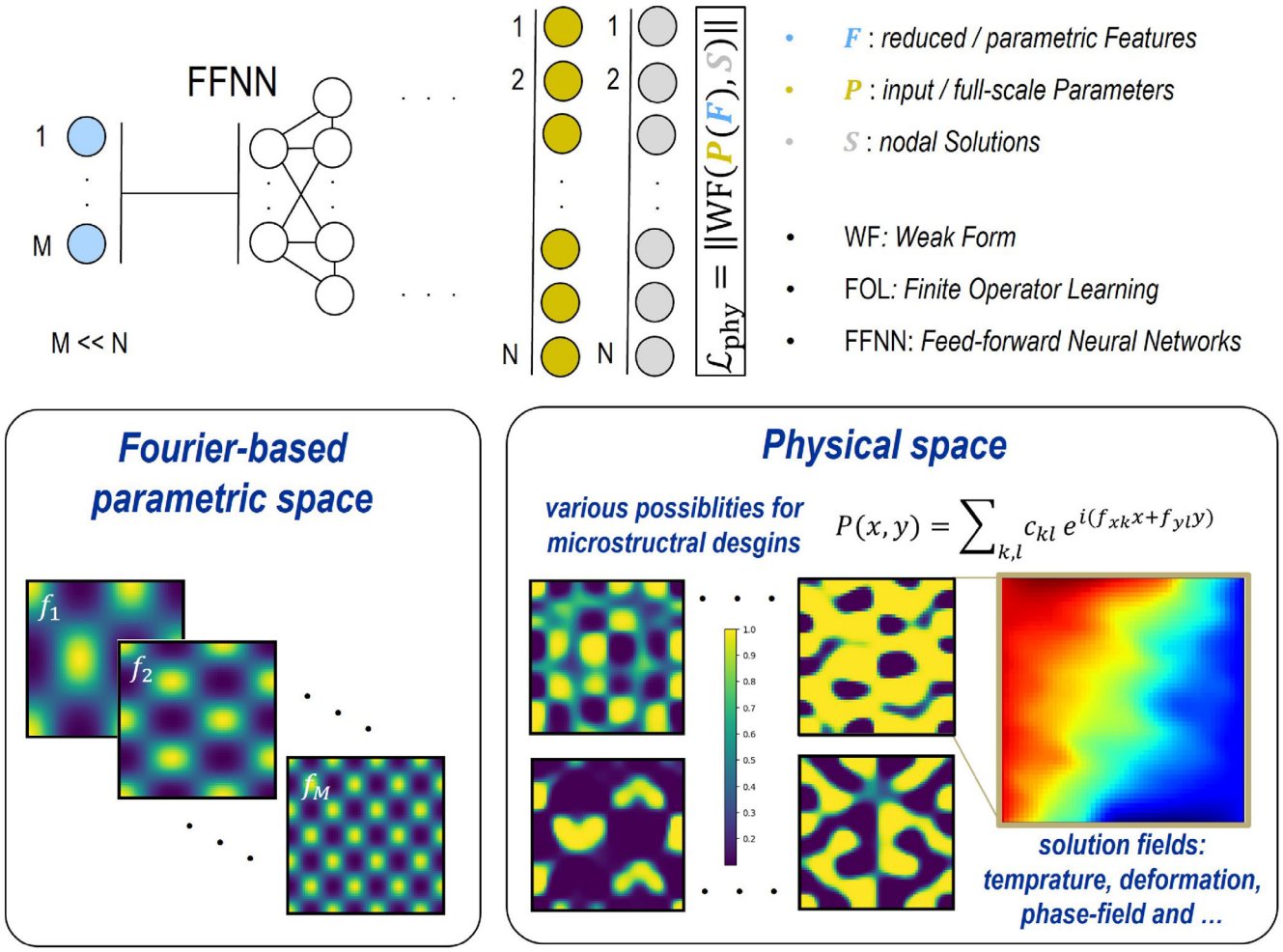


FIGURE 15 | The idea behind the Fourier-based parameterization for FOL is to decrease the input parametric space dimension and increase the resolution density of the output layer.

$$\boldsymbol{X} = \left[A_i, B_i, C_i, D_i\right], \quad \boldsymbol{Y} = \left[U_j, V_j\right], \quad i = 1 \cdots M, \quad j = 1 \cdots N \tag{20}$$

$$\boldsymbol{Y} = \mathcal{N}_{ff}(\boldsymbol{X}; \boldsymbol{\theta}), \quad \boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{b}\} \tag{21}$$

For the rest of the study, and for simplification purposes, only the constant term as well as the last term involving the multiplication of two cosine functions are considered (i.e., $A_i = 0, B_i = 0, C_i = 0$). However, even by limiting ourselves to these terms, one can produce many designs for the property function E_f. We select three frequencies for each direction. Considering the constant term, this choice gives us $M = 3 \times 3 + 1 = 10$ different terms, which can be added up to construct $E(x, y)$ according to Equations (19) and (18).

The input samples used for training are generated by combining different random distributions for these ten inputs. A summary of all the 2000 selected samples is shown in Figure 16. See also Table 3 for a summary of the Fourier-based parameterization for FOL model, as well as Figure B3 for a few examples of samples obtained by the Fourier-based parameterization.

Few unseen test cases and their corresponding Fourier coefficients are summarized in Table 4. These cases are chosen in a way to examine if the proposed methodology can handle different possible scenarios. For example, in some cases, the matrix material is soft and reinforced by hard inclusions, while in others, the matrix is hard and includes soft parts (similar to voids). It should be mentioned that for a given image of the microstructure, using the Fourier-based parameterization for FOL model, one must first perform an inverse Fourier analysis to find the corresponding coefficients for the input layer, which is a straightforward task.

Thanks to the new design of the FOL, we increase the final mesh resolution and avoid additional costs for training. We have checked this matter by testing the idea for different choices of N as shown in Figure 17. Not only do the training and evaluation times (or costs) remain almost unchanged by dramatically increasing the neurons in the output layer from 121 to 2061, but we also observe a mesh convergence in the results. All the training is performed using exactly the same hyperparameters for the deep learning model to ensure a fair comparison. However, based on the given resolution for the output layer, we can tune each neural network more effectively. The latter point is beyond the scope of the current study. Moreover, we observe significant reduction in evaluation cost as we go to higher mesh densities. For the FEM calculations we do not claim that this implementation is the fastest available one.

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

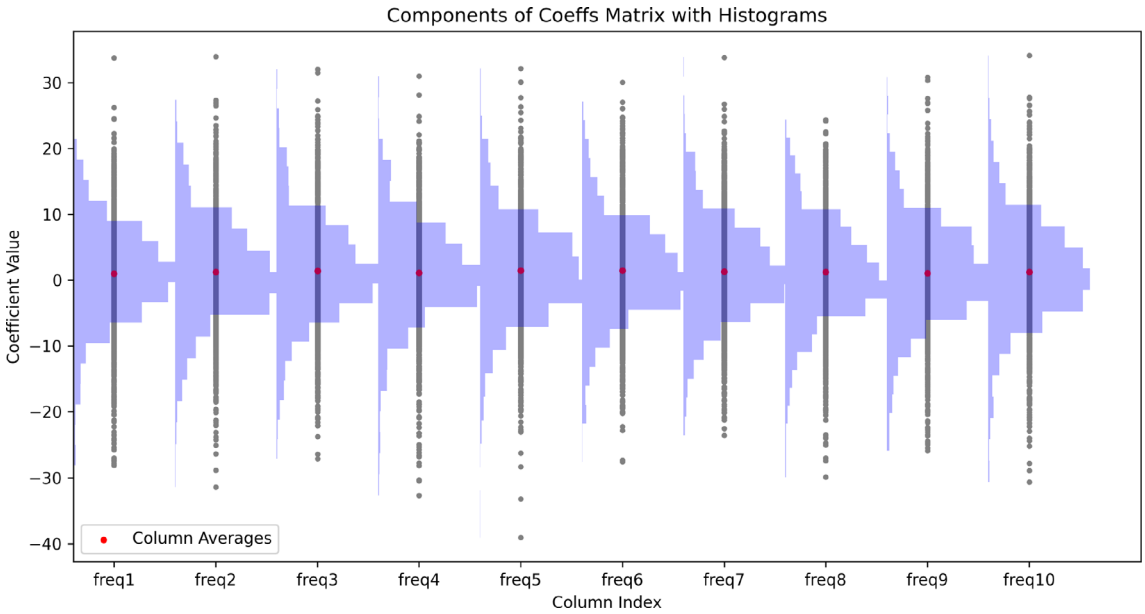


FIGURE 16 | Distribution of the 2000 random values for components of each frequency term. The parameter freq_i represents the ith coefficient corresponding to the ith frequency.

TABLE 3 | Parameters used for generating elasticity samples in training the Fourier-based FOL.

Parameter	Value
Frequency in the x-direction $f_{x,i}$	$\{3, 5, 7\}$
Frequency in the y-direction $f_{y,i}$	$\{2, 4, 7\}$
Number of random samples, batch size	2000, 1
Neurons in the input layer (input features, freq_i)	10
Mesh (neurons in the output layer without DBCs)	11×11 (242), 21×21 (882), 51×51 (5202)
Act. func., hidden layers, learning rate, epoch num.	Swish, [300, 300], 0.001, 5000

TABLE 4 | Selected test cases and corresponding Fourier coefficients.

Study type	Value ($\text{freq}_i, i = 1 \cdots 10$)
Mesh study, Figure 17	[−4.0,2.0, −0.6,1.7,6.0,1.9,10.8, −3.6, −2.8, −9.9]
Microstructure with voids, Figures 18 and 19	[12.7,6.1, −2.1, −14.1, −4.6,2.1,0.4,2.7,4.4, −3.1]
Microstructure with inclusions, Figure 20	[−6.3, −0.4,9.5,4.4,2.2,0.9,1.1,0.4, −3.2,0.3]
Random microstructure, Figure 21	[15.9,17.2,14.3,2.0,13.6,2.5, −6.0,7.4, −7.3,8.2]

The results reported in Figures 18–21 show an acceptable agreement between the two methods. All these results are obtained for the 51×51 and no interpolation or any other additional post-processing step is done for the reported results. Very similar behavior is also observed for other mesh densities.

4.7 | Comparisons with PINN Based Methods Such as DEM

In the FOL framework, if we consider one sample as an input and set the batch size for the deep learning model to one, we end up with the exact formulation for FEM. The only difference is that we minimize the loss using the ADAM or L-BFGS algorithm (i.e., an iterative solver) instead of the standard stiffness or tangent matrix in FEM (i.e., a direct solver). In this particular case, where we are not seeking parametric learning and solely looking for the solution of one specific BVP, the FOL approach is then similar to the PINN or, more accurately, to the DEM approach. However, instead of using automatic differentiation, we use here shape functions to calculate the derivatives. Moreover, concepts from domain decomposition and node connectivity in FEM are inherently integrated into the loss function formulation, which provides additional benefits. To evaluate its more accurate predictions quantitatively, we compare the FOL against three PINN-based formulations, using the FEM solution as the reference solution.

The adopted BVP is already discussed in [21]. The geometry, boundary conditions, as well as the location of collocation points and finite element meshes, are described in Figure 22. For the FOL training, we use only 1 neuron and 1 layer, with the Swish activation function. The entire training takes about 1000 epochs and only a few seconds, even on personal laptops with CPU power. The results for the displacement and stress component in the x direction are reported in Figure 23, where we observe the superiority of FOL over the other methods. What is noteworthy

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

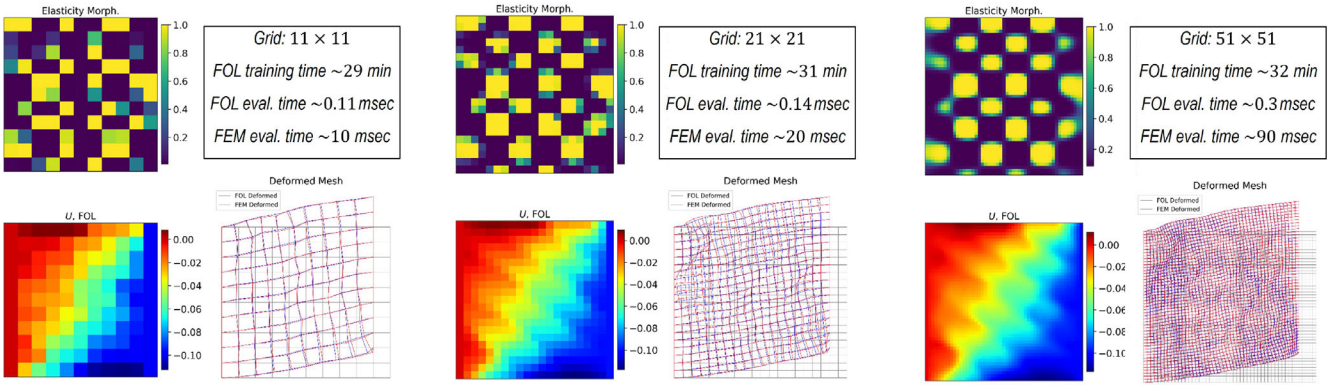


FIGURE 17 | Comparison of the results from FEM and Fourier-based parameterization for FOL. Convergence of the results with respect to the number of elements is demonstrated. The real-time computational cost using a Quadro RTX 6000 GPU for deep learning and CPU Ryzen pro7 for the FEM using FEAP software [65].

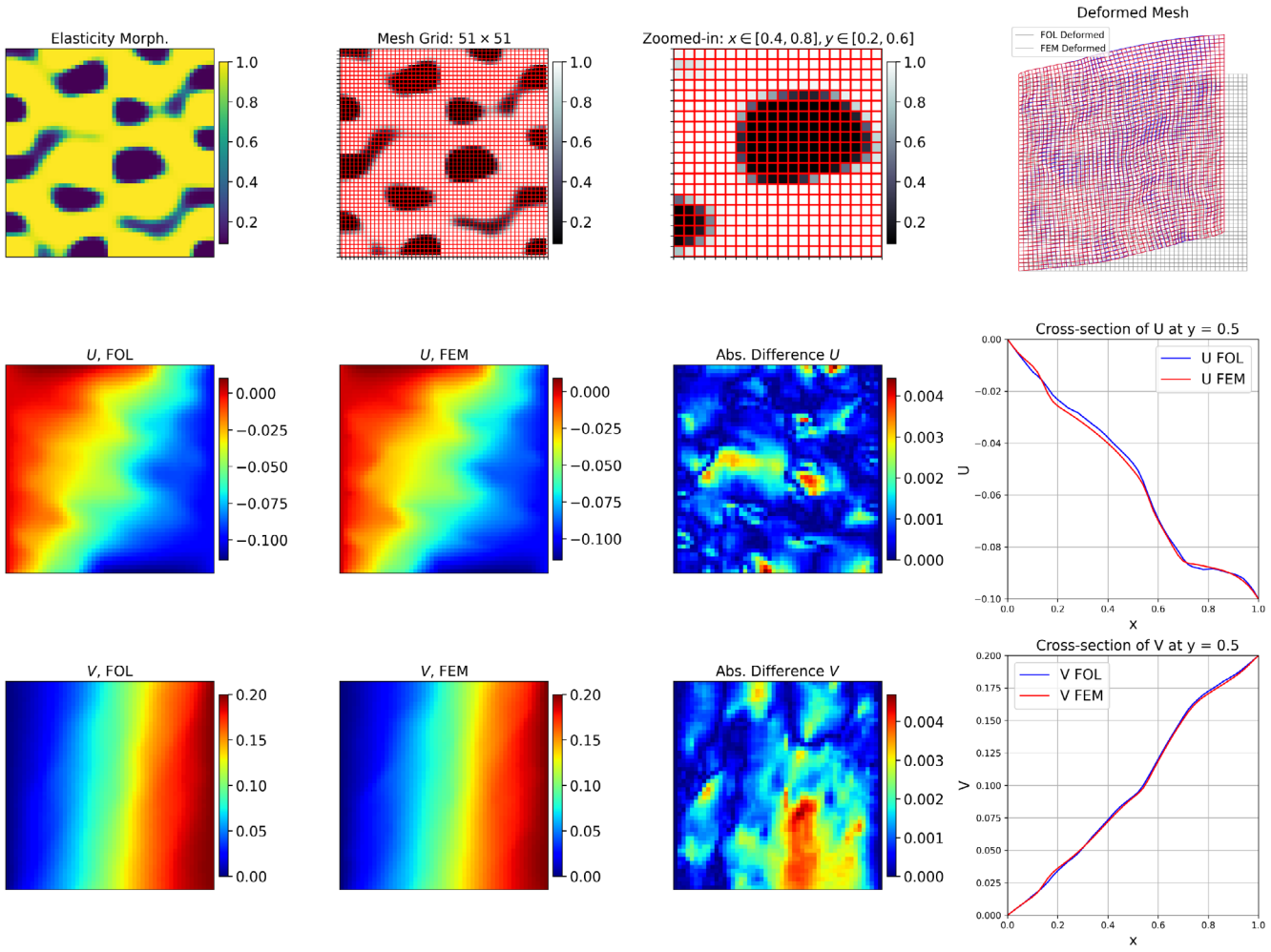


FIGURE 18 | Comparison of the deformation profile between FOL and FEM for a random microstructure with voids by using the Fourier coefficients [12.7,6.1, −2.1, −14.1, −4.6,2.1,0.4,2.7,4.4, −3.1].

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

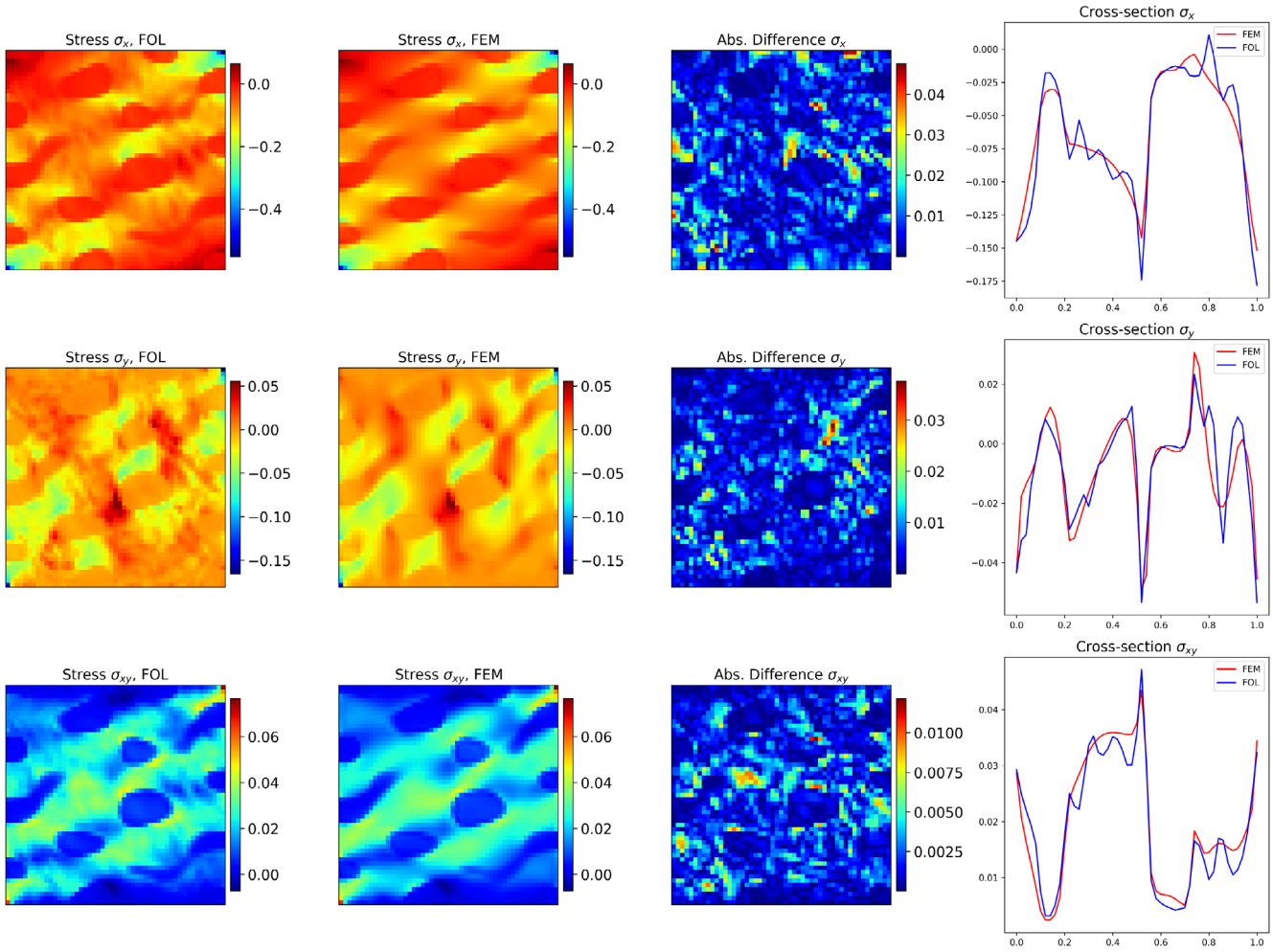


FIGURE 19 | Comparison of the stress profile between FOL and FEM for a random microstructure by using the Fourier coefficients [12.7,6.1, −2.1, −14.1, −4.6,2.1,0.4,2.7,4.4, −3.1].

is that the training time for FOL is significantly lower compared with the other methods, with much fewer required epochs (about 100 times fewer epochs).

4.8 | Extension to 3D

We now intend to apply the FOL framework as a standalone solver to further explore its benefits in a 3D setting. Even in this case, despite the slower convergence rate, the FOL approach offers two significant benefits. First, for some nonlinear problems, this aspect might be interesting since one can bypass the linearization part of the residuals. This aspect remains to be demonstrated in future developments. Second, our investigations indicate that by avoiding the construction of the stiffness matrix, we can save a significant amount of memory. This memory efficiency allows us to achieve much higher mesh densities with the same hardware and implementation setup. For instance, with the aforementioned CPU and GPU capabilities, we were unable to perform FEM analysis using a $50 \times 50 \times 50$ mesh and FEAP software, while the FOL computation was completed successfully.

To show the applicability to model 3D microstructures, we investigated the following 3D example. The framework is again based on the Fourier-based FOL, and this time, three sets of frequencies are considered for the x, y, and z directions (see also Table 5). For the sample shown on the right hand side of Figure 24, we compare the results from the FEM and FOL in left side of Figure 24. Note that in this study, we used unstructured tetrahedral elements for the discretization, which are known to be very flexible to enmesh complex geometries. For this example Gmsh [66] is used. Finally, the following Drichlet BCs are specified: the back surface fully fixed while the front surface is pulled via $\left[U_R = 0.5, V_R = -0.5, W_R = 0.0\right]$.

Consider now a given 3D microstructure with fixed inclusions. We now try to train it based on parametric boundary conditions. In this setup, we use the so-called *Neovius* microstructure [67], with a surface equation based on $3(\cos(x) + \cos(y) + \cos(z)) + 4\cos(x)\cos(y)\cos(z)$, as shown in Figure 25. This shape is motivated by research in the field of meta-materials, and the goal is to predict the mechanical response under any given displacement vector.

In the lower part of Figure 25, the results of the deformed microstructure under two different unseen loading cases are shown, where we can observe a very acceptable agreement in deformation magnitude between the FOL and FEM. For this

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

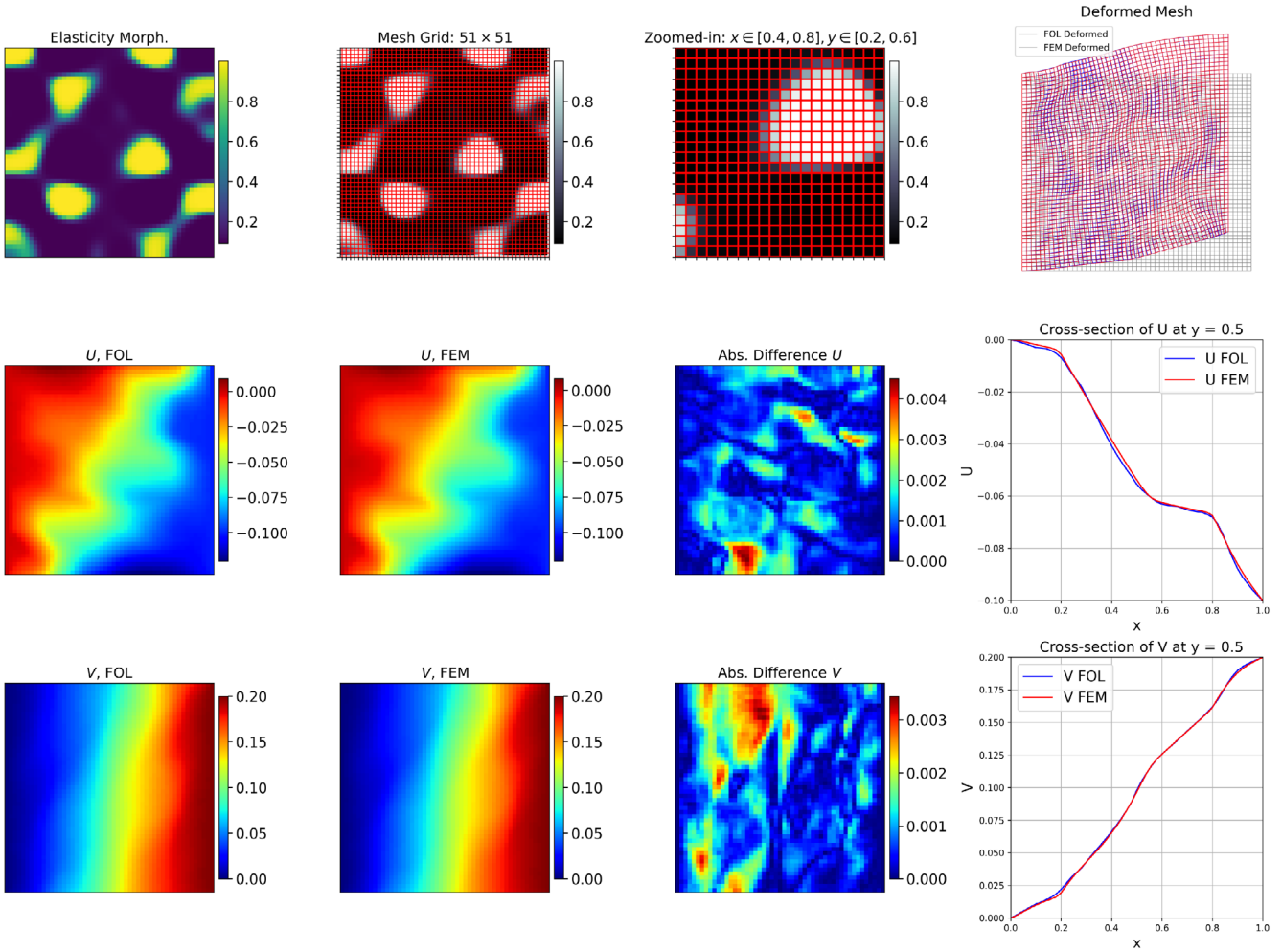


FIGURE 20 | Comparison of the deformation profile between FOL and FEM for a random microstructure with inclusions by using the Fourier coefficients [−6.3, −0.4,9.5,4.4,2.2,0.9,1.1,0.4, −3.2,0.3].

study, we use 2 layers with 10 neurons in each. The batch size is set to 1, and the swish activation function is used. The learning rate is set to 0.0005, and the model is trained for 1000 epochs. The training samples are the displacement components on the front surface, and we use a normalized, randomly generated algorithm to construct the training field. After this successful training, new aspects and opportunities open up for fast multiscale analysis, potentially speeding up classical FE^2 algorithms, since we can access the full solution under a given macroscopic strain tensor.

5 | Conclusion and Outlooks

This work presents a data-free, physics-based method for parametric learning of partial differential equations in the field of mechanics of heterogeneous materials. This method relies solely on governing physical constraints, which are the discretized weak form of the problem as well as Dirichlet boundary terms. Leveraging the FEM without depending on automatic differentiation for computing physical loss functions, the proposed approach approximates all derivatives using shape functions. This significantly improves training efficiency, necessitating fewer epochs and less time per epoch. Additionally, it allows flexibility in selecting activation functions without worrying about vanishing gradients. However, it is important to mention that potential discretization errors from the FEM persist in the present approach. The introduced FOL framework can be seen as an extension of FEM and DEM, providing the ability to parametrically learn any interesting features (inputs) of the given BVP. Moreover, the application of hard boundary conditions in the proposed framework is extremely straightforward, very similar to available methods in FEM.

In this study, the network takes the elasticity distribution within the microstructure as input and produces the deformation components under fixed boundary conditions as output. Our method demonstrates accurate predictions of deformation fields, with the maximum pointwise error across all test cases consistently below 10%, primarily localized near phase boundaries. On the other hand, errors for first-order gradient terms, such as stress components, exhibit higher values. This observation requires further improvement in future developments, as local stress values significantly affect the localized nonlinear material behavior, such as plasticity and damage evolution. On the positive side, errors for averaged or homogenized values remain around 1%, making them acceptable for various engineering applications as long as the material is in the elastic regime. Despite possible errors and

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

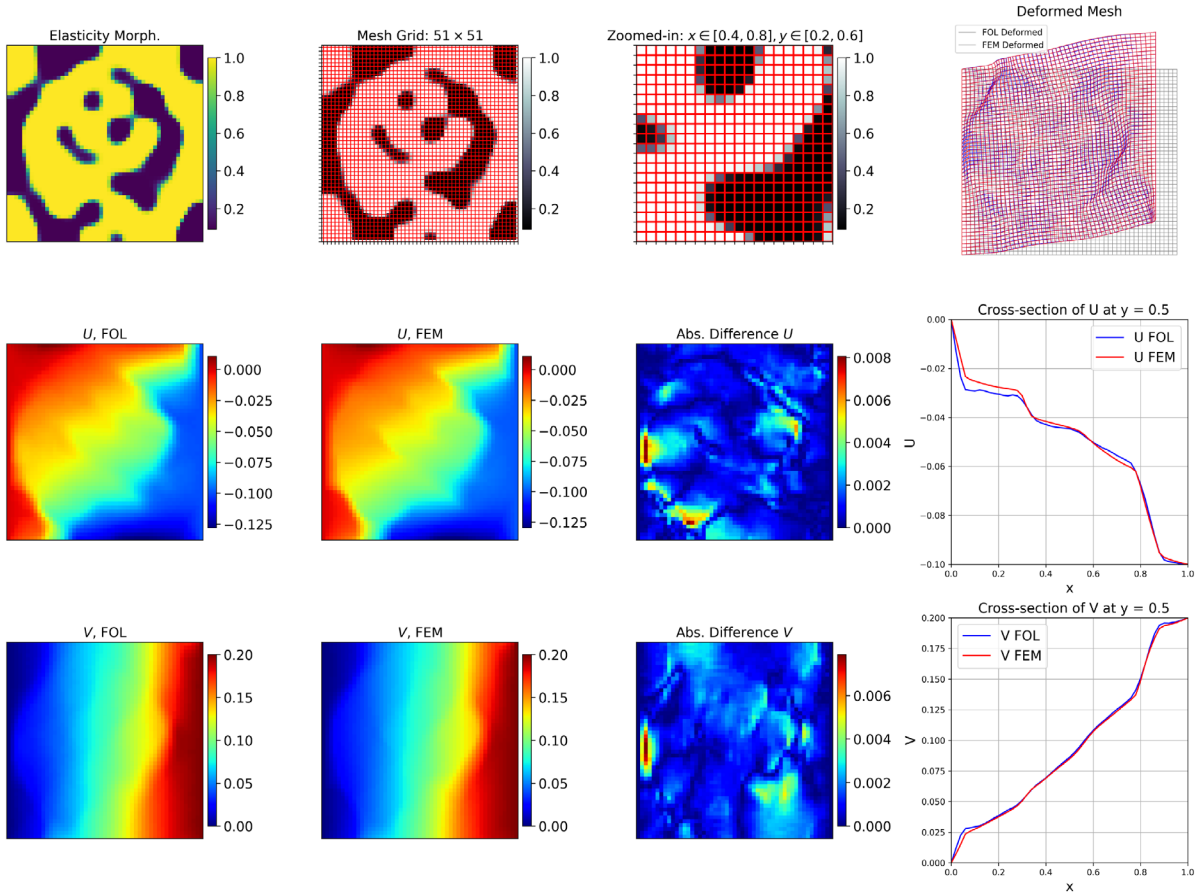


FIGURE 21 | Comparison of the deformation profile between FOL and FEM for a random microstructure by using following set of Fourier coefficients [15.9,17.2,14.3,2.0,13.6,2.5, −6.0,7.4, −7.3,8.2].

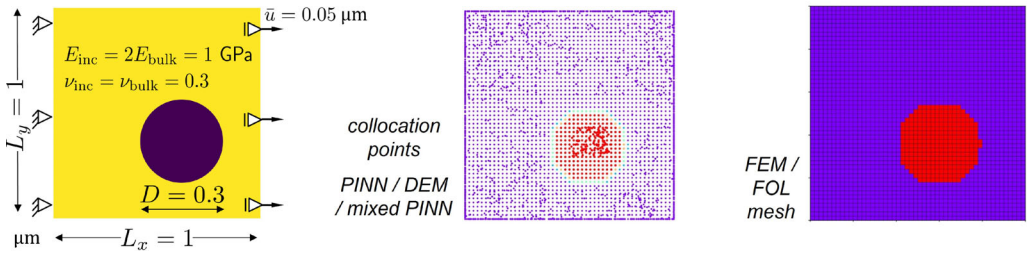


FIGURE 22 | Geometry and BCs for comparison of different models. See also [21].

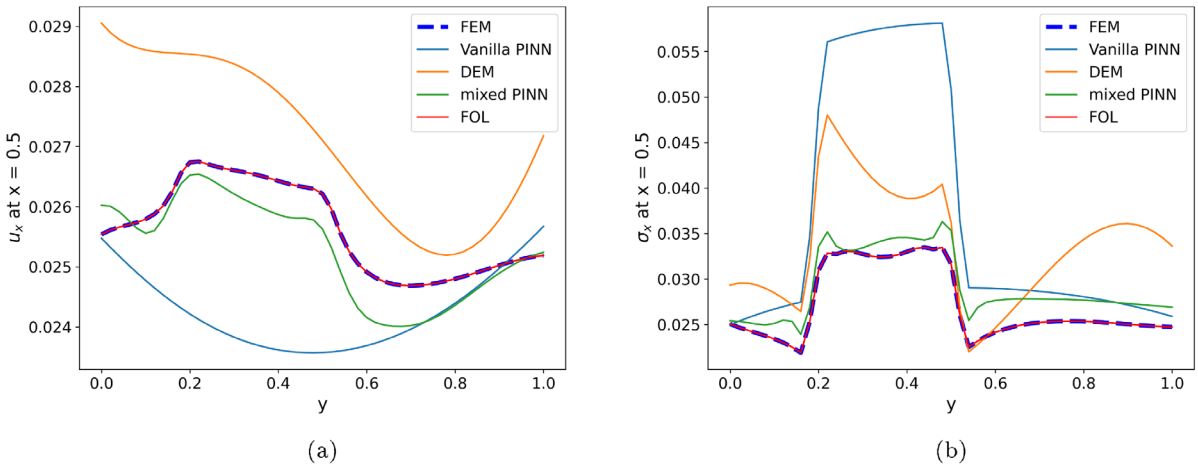


FIGURE 23 | Comparison the results obtained from the FEM, FOL, and other PINN-based models.

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

deviations in predicting the solution, the introduced methodology offers a significant speedup over the standard FEM in elasticity. Furthermore, compared with a data-driven approach, the introduced physics-driven method demonstrates superior accuracy for unseen input parameters. We also discussed two different approaches to reduce the input parametric space while still achieving high-resolution solutions. The first approach is based on the MEA, which transfers the low-resolution solution to higher grid points through a pretrained autoencoder. The advantage of this approach is that any complex microstructure shape can be fed into the model to be reduced, solved, and upscaled. The potential drawback is that the training process of the autoencoder requires high-resolution FEM data. The second approach leverages Fourier-based parameterization for deep learning, where the input layer is significantly reduced to the coefficients of selected frequencies of sine and cosine functions in different spatial directions. The advantage of this approach is that we do not require any additional labeled data and can easily obtain the solution on much denser grids without any significant change in training time. However, obtaining the correct frequency coefficients for an arbitrary complex microstructure shape requires an additional step of an inverse Fourier transform. Both methods demonstrated excellent performance in predicting deformation patterns. The prediction of spatial derivatives, and therefore stress values, are also in good agreement, especially in the averaged sense. Ideas on how to improve them are discussed as well.

TABLE 5 | Parameters used for generating the elasticity sample for the 3D Fourier-based FOL.

Parameter	Value
Frequency in the x-direction $f_{x,i}$	$\{2, 4, 6\}$
Frequency in the y-direction $f_{y,i}$	$\{2, 4, 6\}$
Frequency in the z-direction $f_{z,i}$	$\{2, 4, 6\}$
Number of random samples and batch size	1
Number of neurons in the input layer (input features, freq_i)	28
Number of neurons in the output layer	6920
Activation func., hidden layers, learning rate, epoch	Swish, [1], 0.001, 500

The current investigations open up an entirely new approach for further research into combining physics-based operator learning and well-established numerical methods. Expanding this approach to nonlinear problems is essential, as it holds the potential for even greater speed improvements. Further extensions to more complex geometries are also of great interest. FEM supports arbitrary shapes in structural analysis, which enables the FOL framework to be trained for complex shapes, including those with unstructured meshes. To apply FOL to different geometries, one needs to provide a suitable parameter space to describe the outer shape. For example, CAD-based control points and techniques from isogeometric analysis can be integrated into the FOL framework. While our focus was here on elastic problems in heterogeneous materials, exploring other types of equations, such as temporal problems and microstructure modeled by the phase field method presents an intriguing avenue for our future scope. Finally, it would be interesting to apply the current approach to

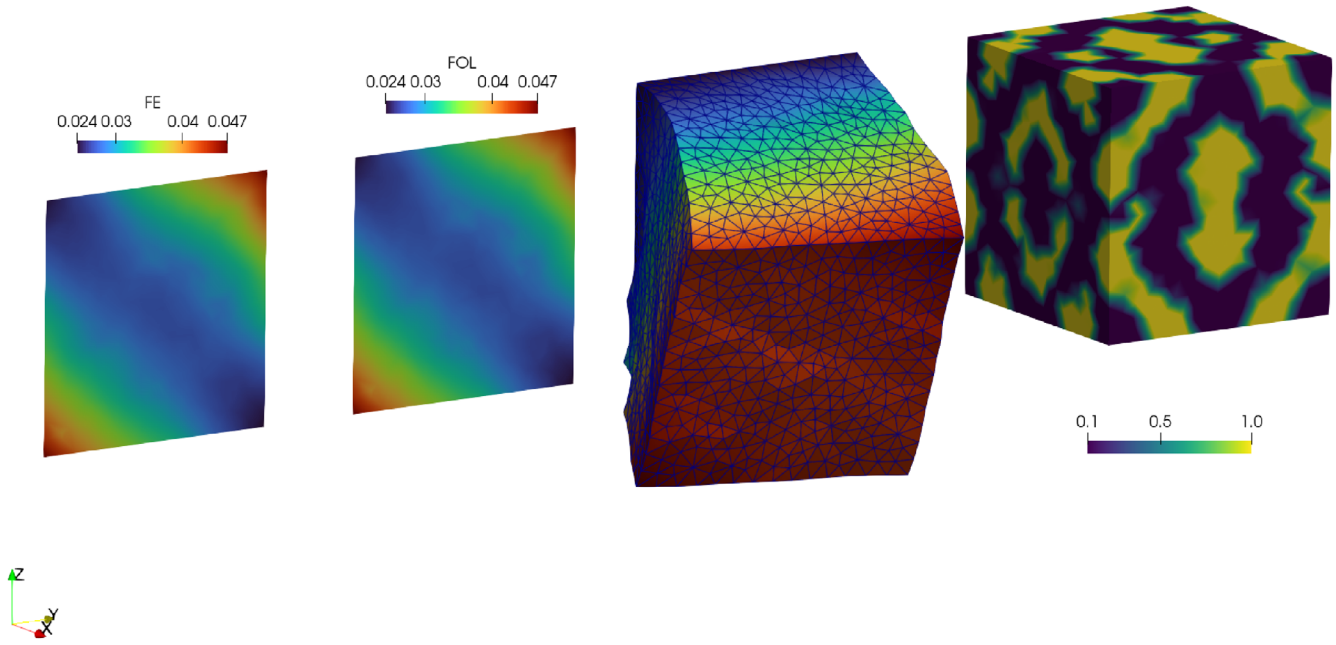


FIGURE 24 | Comparison of the displacement results from FOL and FEM using unstructured tetrahedral elements.

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek, 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

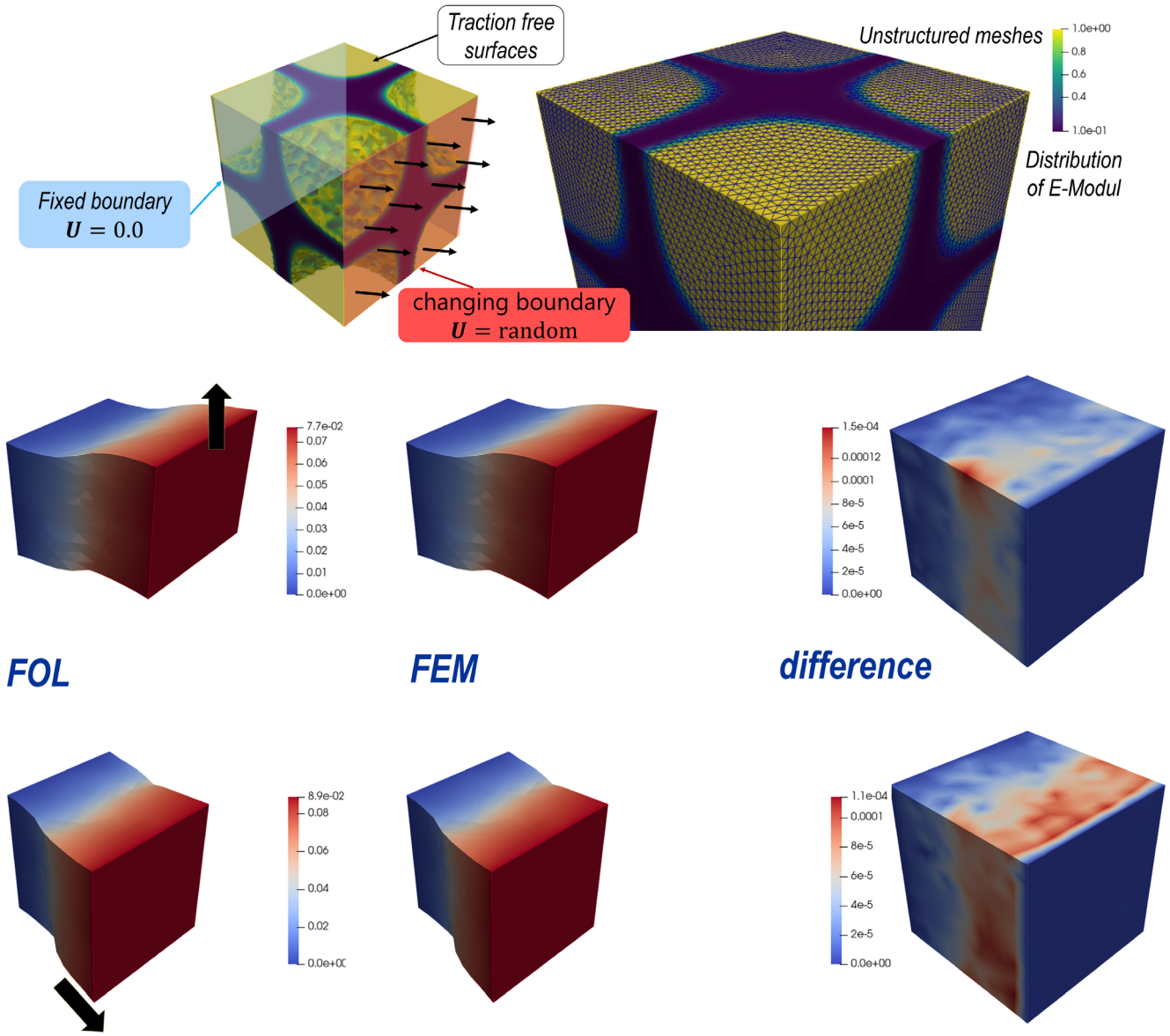


FIGURE 25 | Comparison of the results from FOL (left hand side) and FEM (middle) for a 3D example using parametric learning on the imposed boundary displacements.

inverse problems, wherein providing sufficient data can lead to discovering the underlying physics of the problem.

Author Contributions

Shahed Rezaei: conceptualization, methodology, software, writing – review and editing. **Reza Najian Asl:** methodology, software, review and editing. **Shirko Faroughi:** supervision, review and editing. **Mahdi Asgharzadeh:** software, writing – review and editing. **Ali Harandi:** software, writing – review and editing. **Rasoul Najafi Koopas:** software, review and editing. **Gottfried Laschet:** supervision, review and editing. **Stefanie Reese:** supervision, review and editing. **Markus Apel:** funding, supervision, review and editing.

Acknowledgments

The authors would like to thank the Deutsche Forschungsgemeinschaft (DFG) for the funding support provided to develop the present work in the project Cluster of Excellence "Internet of Production" (project: 390621612). The Authors also acknowledge the financial support of SFB 1120 B07 - Mehrskalige thermomechanische Simulation der fest-flüssig Interaktionen bei der Erstarrung (B07) (260070971) (260070971). Ali Harandi and Stefanie Reese acknowledge the financial support of Transregional Collaborative Research Center SFB/TRR 339 with project number 453596084 funded by DFG gratefully. The author Rasoul Najafi Koopas would like to thank the Zentrum für Digitalisierungs- und Technologieforschung der Bundeswehr (dtec.bw) for their financial support. Shahed Rezaei would like to express his gratitude to Kianoosh Taghikhani for his technical software support that contributed to this work.

Data Availability Statement

The data supporting the findings of this study are openly available and can be accessed via the following link: https://github.com/RezaNajian/FOL/tree/main.

References

1. M. G. D. Geers, V. G. Kouznetsova, and W. A. M. Brekelmans, "Multi-Scale Computational Homogenization: Trends and Challenges,"

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Journal of Computational and Applied Mathematics 234, no. 7 (2010): 2175–2182.

2. S. A. Faroughi, N. M. Pawar, C. Fernandes, et al., "Physics-Guided, Physics-Informed, and Physics-Encoded Neural Networks and Operators in Scientific Computing: Fluid and Solid Mechanics," *Journal of Computing and Information Science in Engineering* 24, no. 4 (2024): 040802.

3. W. K. Liu, S. Li, and H. S. Park, "Eighty Years of the Finite Element Method: Birth, Evolution, and Future," *Archives of Computational Methods in Engineering* 29, no. 6 (2022): 4431–4453.

4. L. Herrmann and S. Kollmannsberger, "Deep Learning in Computational Mechanics: A Review," *Computational Mechanics* 74 (2024): 281–331.

5. D. Kim and J. Lee, "A Review of Physics Informed Neural Networks for Multiscale Analysis and Inverse Problems," *Communications Physics* 6 (2024): 1–11.

6. L. Lu, P. Jin, G. Pang, Z. Zhang, and G. Karniadakis, "Learning Nonlinear Operators via Deeponet Based on the Universal Approximation Theorem of Operators," *Nature Machine Intelligence* 3 (2021): 218–229.

7. Z. Li, N. Kovachki, K. Azizzadenesheli, et al., "Fourier Neural Operator for Parametric Partial Differential Equations," arXiv:2010.08895, 2021.

8. Z. Li, N. Kovachki, K. Azizzadenesheli, et al., "Neural Operator: Graph Kernel Network for Partial Differential Equations," arXiv:2003.03485, 2020.

9. G. Chen, L. Xu, Q. Meng, C. Lu, C. Liu, and Y. Li, "Learning Neural Operators on Riemannian Manifolds," arXiv:2302.08166, 2023.

10. Q. Cao, S. Goswami, and G. E. Karniadakis, "Lno: Laplace Neural Operator for Solving Differential Equations," arXiv:2303.10528, 2023.

11. L. Lu, X. Meng, S. Cai, et al., "A Comprehensive and Fair Comparison of Two Neural Operators (With Practical Extensions) Based on Fair Data," *Computer Methods in Applied Mechanics and Engineering* 393 (2022): 114778.

12. M. M. Rashid, S. Chakraborty, and N. M. Anoop Krishnan, "Revealing the Predictive Power of Neural Operators for Strain Evolution in Digital Composites," *Journal of the Mechanics and Physics of Solids* 181 (2023): 105444.

13. S. Koric, A. Viswantah, D. W. Abueidda, N. A. Sobh, and K. Khan, "Deep Learning Operator Network for Plastic Deformation With Variable Loads and Material Properties," *Engineering with Computers* 40 (2023): 917–929.

14. M. Lu, A. Mohammadi, Z. Meng, X. Meng, G. Li, and Z. Li, "Deep Neural Operator for Learning Transient Response of Interpenetrating Phase Composites Subject to Dynamic Loading," *Computational Mechanics* 72 (2023): 563–576.

15. J. He, S. Koric, S. Kushwaha, J. Park, D. Abueidda, and I. Jasiuk, "Novel Deeponet Architecture to Predict Stresses in Elastoplastic Structures With Variable Complex Geometries and Loads," *Computer Methods in Applied Mechanics and Engineering* 415 (2023): 116277.

16. J. He, S. Kushwaha, J. Park, S. Koric, D. Abueidda, and I. Jasiuk, "Sequential Deep Operator Networks (s-Deeponet) for Predicting Full-Field Solutions Under Time-Dependent Loads," *Engineering Applications of Artificial Intelligence* 127 (2024): 107258.

17. S. Goswami, M. Yin, Y. Yue, and G. E. Karniadakis, "A Physics-Informed Variational Deeponet for Predicting Crack Path in Quasi-Brittle Materials," *Computer Methods in Applied Mechanics and Engineering* 391 (2022): 114587.

18. I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial Neural Networks for Solving Ordinary and Partial Differential Equations," *IEEE Transactions on Neural Networks* 9, no. 5 (1998): 987–1000.

19. J. Sirignano and K. Spiliopoulos, "Dgm: A Deep Learning Algorithm for Solving Partial Differential Equations," *Journal of Computational Physics* 375 (2018): 1339–1364.

20. M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations," *Journal of Computational Physics* 378 (2019): 686–707.

21. S. Rezaei, A. Harandi, A. Moeineddin, B.-X. Xu, and S. Reese, "A Mixed Formulation for Physics-Informed Neural Networks as a Potential Solver for Engineering Problems in Heterogeneous Domains: Comparison With Finite Element Method," *Computer Methods in Applied Mechanics and Engineering* 401 (2022): 115616.

22. E. Haghighat, M. Raissi, A. Moure, H. Gomez, and R. Juanes, "A Physics-Informed Deep Learning Framework for Inversion and Surrogate Modeling in Solid Mechanics," *Computer Methods in Applied Mechanics and Engineering* 379 (2021): 113741.

23. J. Wu, J. Jiang, Q. Chen, G. Chatzigeorgiou, and F. Meraghni, "Deep Homogenization Networks for Elastic Heterogeneous Materials With Two- and Three-Dimensional Periodicity," *International Journal of Solids and Structures* 284 (2023): 112521.

24. A. M. Roy, R. Bose, V. Sundararaghavan, and R. Arróyave, "Deep Learning-Accelerated Computational Framework Based on Physics Informed Neural Network for the Solution of Linear Elasticity," *Neural Networks* 162 (2023): 472–489.

25. S. Rezaei, A. Moeineddin, and A. Harandi, "Learning Solutions of Thermodynamics-Based Nonlinear Constitutive Material Models Using Physics-Informed Neural Networks," *Computational Mechanics* 74 (2024): 333–366.

26. E. Haghighat, S. Abouali, and R. Vaziri, "Constitutive Model Characterization and Discovery Using Physics-Informed Deep Learning," *Engineering Applications of Artificial Intelligence* 120 (2023): 105828.

27. E. Samaniego, C. Anitescu, S. Goswami, et al., "An Energy Approach to the Solution of Partial Differential Equations in Computational Mechanics via Machine Learning: Concepts, Implementation and Applications," *Computer Methods in Applied Mechanics and Engineering* 362 (2020): 112790.

28. V. M. Nguyen-Thanh, X. Zhuang, and T. Rabczuk, "A Deep Energy Method for Finite Deformation Hyperelasticity," *European Journal of Mechanics – A/Solids* 80 (2020): 103874.

29. V. M. Nguyen-Thanh, C. Anitescu, N. Alajlan, T. Rabczuk, and X. Zhuang, "Parametric Deep Energy Approach for Elasticity Accounting for Strain Gradient Effects," *Computer Methods in Applied Mechanics and Engineering* 386 (2021): 114096.

30. D. W. Abueidda, S. Koric, R. A. Al-Rub, C. M. Parrott, K. A. James, and N. A. Sobh, "A Deep Learning Energy Method for Hyperelasticity and Viscoelasticity," *European Journal of Mechanics – A/Solids* 95 (2022): 104639.

31. J. N. Fuhg and N. Bouklas, "The Mixed Deep Energy Method for Resolving Concentration Features in Finite Strain Hyperelasticity," *Journal of Computational Physics* 451 (2022): 110839.

32. A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, "Conservative Physics-Informed Neural Networks on Discrete Domains for Conservation Laws: Applications to Forward and Inverse Problems," *Computer Methods in Applied Mechanics and Engineering* 365 (2020): 113028.

33. E. Kharazmi, Z. Zhang, and E. M. K. George, "hp-vpinns: Variational Physics-Informed Neural Networks With Domain Decomposition," *Computer Methods in Applied Mechanics and Engineering* 374 (2021): 113547.

34. L. McClenny and U. Braga-Neto, "Self-Adaptive Physics-Informed Neural Networks Using a Soft Attention Mechanism," arXiv:2009.04544, 2020.

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

35. S. Faroughi, A. Darvishi, and S. Rezaei, "On the Order of Derivation in the Training of Physics-Informed Neural Networks: Case Studies for Non-Uniform Beam Structures," *Acta Mechanica* 234 (2023): 5673–5695.

36. S. Wang, S. Sankaran, H. Wang, and P. Perdikaris, "An Expert's Guide to Training Physics-Informed Neural Networks," arXiv:2308.08468, 2023.

37. Z. Li, H. Zheng, N. Kovachki, et al., "Physics-Informed Neural Operator for Learning Partial Differential Equations," arXiv:2111.03794, 2023.

38. S. Wang, H. Wang, and P. Perdikaris, "Learning the Solution Operator of Parametric Partial Differential Equations With Physics-Informed Deeponets," *Science Advances* 7, no. 40 (2021): eabi8605.

39. Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris, "Physics-Constrained Deep Learning for High-Dimensional Surrogate Modeling and Uncertainty Quantification Without Labeled Data," *Journal of Computational Physics* 394 (2019): 56–81.

40. H. Gao, L. Sun, and J.-X. Wang, "Phygeonet: Physics-Informed Geometry-Adaptive Convolutional Neural Networks for Solving Parameterized Steady-State Pdes on Irregular Domain," *Journal of Computational Physics* 428 (2021): 110079.

41. X.-Y. Liu, L. L. Min Zhu, H. Sun, and J.-X. Wang, "Multi-Resolution Partial Differential Equations Preserved Learning Framework for Spatiotemporal Dynamics," *Communications Physics* 7 (2024): 31.

42. Z. Zhang and G. X. Gu, "Physics-Informed Deep Learning for Digital Materials," *Theoretical and Applied Mechanics Letters* 11, no. 1 (2021): 100220.

43. K. Kontolati, S. Goswami, G. E. Karniadakis, and M. D. Shields, "Learning in Latent Spaces Improves the Predictive Accuracy of Deep Neural Operators," 2023.

44. X. Zhang and K. Garikipati, "Label-Free Learning of Elliptic Partial Differential Equation Solvers With Generalizability Across Boundary Value Problems," *Computer Methods in Applied Mechanics and Engineering* 417 (2023): 116214.

45. P. Ren, C. Rao, Y. Liu, J.-X. Wang, and H. Sun, "Phycrnet: Physics-Informed Convolutional-Recurrent Network for Solving Spatiotemporal Pdes," *Computer Methods in Applied Mechanics and Engineering* 389 (2022): 114399.

46. X. Zhao, Z. Gong, Y. Zhang, W. Yao, and X. Chen, "Physics-Informed Convolutional Neural Networks for Temperature Field Prediction of Heat Source Layout Without Labeled Data," *Engineering Applications of Artificial Intelligence* 117 (2023): 105516.

47. Z. Zhang, X. Yan, P. Liu, K. Zhang, R. Han, and S. Wang, "A Physics-Informed Convolutional Neural Network for the Simulation and Prediction of Two-Phase Darcy Flows in Heterogeneous Porous Media," *Journal of Computational Physics* 477 (2023): 111919.

48. J. N. Fuhg, A. Karmarkar, T. Kadeethum, H. Yoon, and N. Bouklas, "Deep Convolutional Ritz Method: Parametric Pde Surrogates Without Labeled Data," *Applied Mathematics and Mechanics* 44, no. 7 (2023): 1151–1174.

49. J. He, D. Abueidda, S. Koric, and I. Jasiuk, "On the Use of Graph Neural Networks and Shape-Function-Based Gradient Computation in the Deep Energy Method," *International Journal for Numerical Methods in Engineering* 124, no. 4 (2023): 864–879.

50. S. Rezaei, A. Moeineddin, M. Kaliske, and M. Apel, "Integration of Physics-Informed Operator Learning and Finite Element Method for Parametric Learning of Partial Differential Equations," arXiv:2401.02363, 2024.

51. Y. Yamazaki, A. Harandi, M. Muramatsu, et al., "A Finite Element-Based Physics-Informed Operator Learning Framework for Spatiotemporal Partial Differential Equations on Arbitrary Domains," 2024.

52. T. Chen and H. Chen, "Universal Approximation to Nonlinear Operators by Neural Networks With Arbitrary Activation Functions and Its Application to Dynamical Systems," *IEEE Transactions on Neural Networks* 6, no. 4 (1995): 911–917.

53. S. Wang, H. Wang, and P. Perdikaris, "Improved Architectures and Training Algorithms for Deep Operator Networks," *Journal of Scientific Computing* 92, no. 2 (2022): 35.

54. E. Haghighat, U. B. Waheed, and G. Karniadakis, "En-deeponet: An Enrichment Approach for Enhancing the Expressivity of Neural Operators With Applications to Seismology," *Computer Methods in Applied Mechanics and Engineering* 420 (2024): 116681.

55. M. Gottfried Laschet, M. Abouridouane, M. B. Fernández, and T. Bergs, "Microstructure Impact on the Machining of two gear steels. Part 1: Derivation of Effective Flow Curves," *Materials Science and Engineering A* 845 (2022): 143125.

56. D. Vogiatzief, A. Evirgen, M. Pedersen, and U. Hecht, "Laser Powder Bed Fusion of An Al-cr-fe-ni High-Entropy Alloy Produced by Blending of Prealloyed and Elemental Powder: Process Parameters, Microstructures and Mechanical Properties," *Journal of Alloys and Compounds* 918 (2022): 165658.

57. J. R. Mianroodi, S. Rezaei, N. H. Siboni, B.-X. Xu, and D. Raabe, "Lossless Multi-Scale Constitutive Elastic Relations With Artificial Intelligence," *npj Computational Materials* 8 (2022): 1–12.

58. J. Bradbury, R. Frostig, P. Hawkins, et al., "JAX: Composable Transformations of Python+NumPy Programs," 2018, http://github.com/google/jax.

59. E. Haghighat and R. Juanes, "Sciann: A Keras/Tensorflow Wrapper for Scientific Computations and Physics-Informed Deep Learning Using Artificial Neural Networks," *Computer Methods in Applied Mechanics and Engineering* 373 (2021): 113552.

60. P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for Activation Functions," arXiv:1710.05941, 2017.

61. D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," arXiv:1412.6980, 2017.

62. R. N. Koopas, S. Rezaei, N. Rauter, R. Ostwald, and R. Lammering, "Introducing a Microstructure-Embedded Autoencoder Approach for Reconstructing High-Resolution Solution Field From Reduced Parametric Space," arXiv preprint arXiv:2405.01975, 2024.

63. M. Tancik, P. P. Srinivasan, B. Mildenhall, et al., "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains," 2020.

64. A. Harandi, A. Moeineddin, M. Kaliske, S. Reese, and S. Rezaei, "Mixed Formulation Of Physics-Informed Neural Networks for Thermo-Mechanically Coupled Systems and Heterogeneous Domains," *International Journal for Numerical Methods in Engineering* 125, no. 4 (2024): e7388.

65. R. L. Taylor, "FEAP – Finite Element Analysis Program," 2014.

66. C. Geuzaine and J.-F. Remacle, "Gmsh: A 3-d Finite Element Mesh Generator With Built-In Pre- and Post-Processing Facilities," 2009.

67. J. Feng, F. Jianzhong, X. Yao, and Y. He, "Triply Periodic Minimal Surface (tpms) Porous Structures: From Multi-Scale Design, Precise Additive Manufacturing to Multidisciplinary Applications," *International Journal of Extreme Manufacturing* 4, no. 2 (2022): 022001.

68. N. Boullé and A. Townsend, "A Mathematical Guide to Operator Learning," arXiv:2312.14688, 2023.

Appendix A

Details About the Training of the DeepONet

Here, we provide additional details about the training process of the DeepONet network. Section 3.2 illustrates the vanilla DeepONet network

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

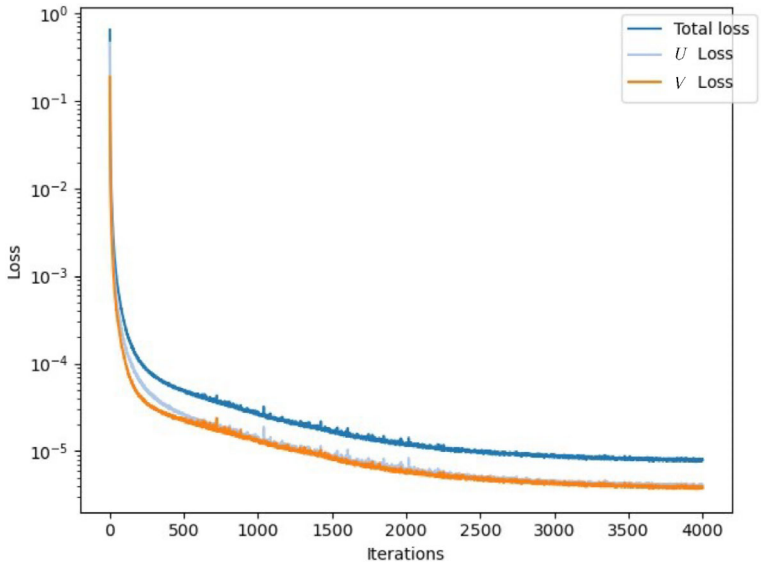


FIGURE A1 | The decay of total loss and its U and V components for the DeepONet network with optimally found hyperparameters.

architecture in Figure 3. The branch network processes the distribution of properties, which are measured at arbitrary sensor locations. Specifically, the input function to the branch network includes the values of Young's modulus at the element nodes. The branch maps the input function into two distinct vectors, one for each output. They can be considered as latent vectors, see [68]. On the other hand, the trunk network is designed to receive the spatial coordinates of a single point where the output function is evaluated. Finally, two operators are built by the dot product of the last layers of the branch and the trunk.

These operators are designed to map the input function $\boldsymbol{E}$, captured at N sensor locations (nodes of elements), to the corresponding solutions at the given spatial coordinate $\boldsymbol{X} = (x_i, y_i)$ to the trunk network.

In Equation (17), the loss function is defined as the mean squared error (MSE) between the predicted values of the operators for U and V and the solutions derived using the finite element method (FEM). For each input function, indexed by i and covering the entire range of training inputs up to I_{train}, the total number of training input functions, the operator predicts the value at each point j. Then the difference between these predictions and the solutions obtained by FEM over N points are computed. The network parameters, denoted as θ, are then optimized by minimizing the loss function outlined in Equation (17).

For training the network, a purely data-driven approach is employed. Details regarding the sample preparation can be found in Section 3.3. The training involves 4000 samples, each featuring distinct morphology maps, and the training is done for 4000 epochs. Since the architecture of the network differs significantly from the FOL approach, an extensive study of hyperparameters was performed to achieve optimal results. The best-performing configuration was found to be 6 [20], which indicates 6 hidden layers when each layer consists of 20 neurons. Notably, both the branch and the trunk networks use an identical architecture and share a common final layer containing 20 neurons.

The best result is obtained by using the *swish* as the activation function. The loss decay for the optimal hyperparameters is shown in Figure A1.

Appendix B

Performance of MEA Approach

In Figure B1, we provide more results on the performance of the MEA approach on different test cases. Note that none of these samples are used in the training of the FOL or in the training process of the MEA approach. As mentioned before, the MEA approach is able to map the low-resolution solution from the FOL (or even from any other solver, if applicable) onto a high-resolution grid space, capturing even sharp jumps in the solution quite effectively. Using the MEA approach will enable users to focus on training their neural operators solely based on the reduced topological space and input parameters, which makes the training efficient and more accurate.

In Figures B2 and B3, few examples of the randomly generated samples for the training of the original FOL and Fourier-based FOL are shown, respectively.

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

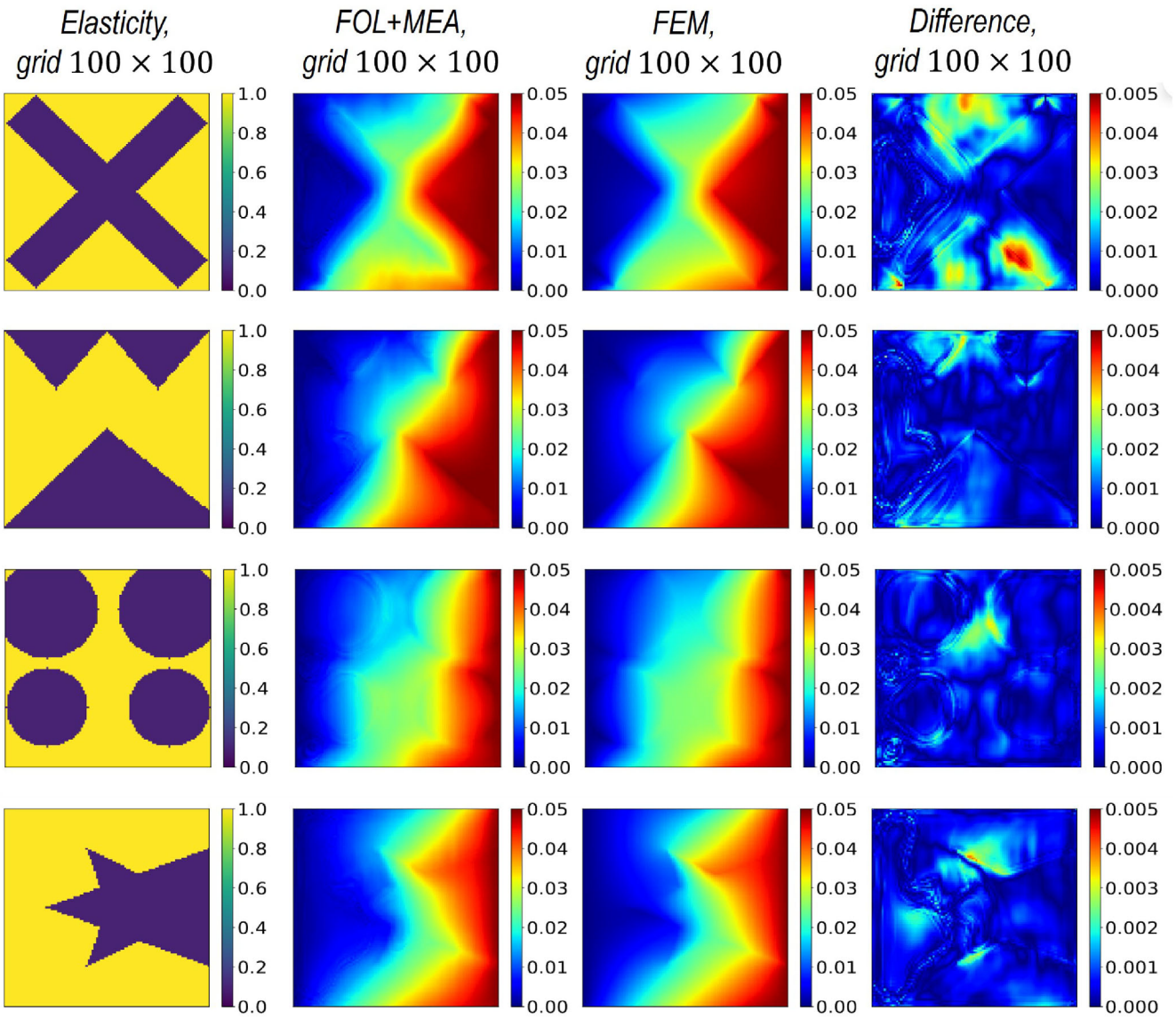


FIGURE B1 | Prediction of the microstructure-embedded autoencoder (MEA) method for the mechanical deformation on high resolution grid (displacement component U evaluated on a 101×101 grid).

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

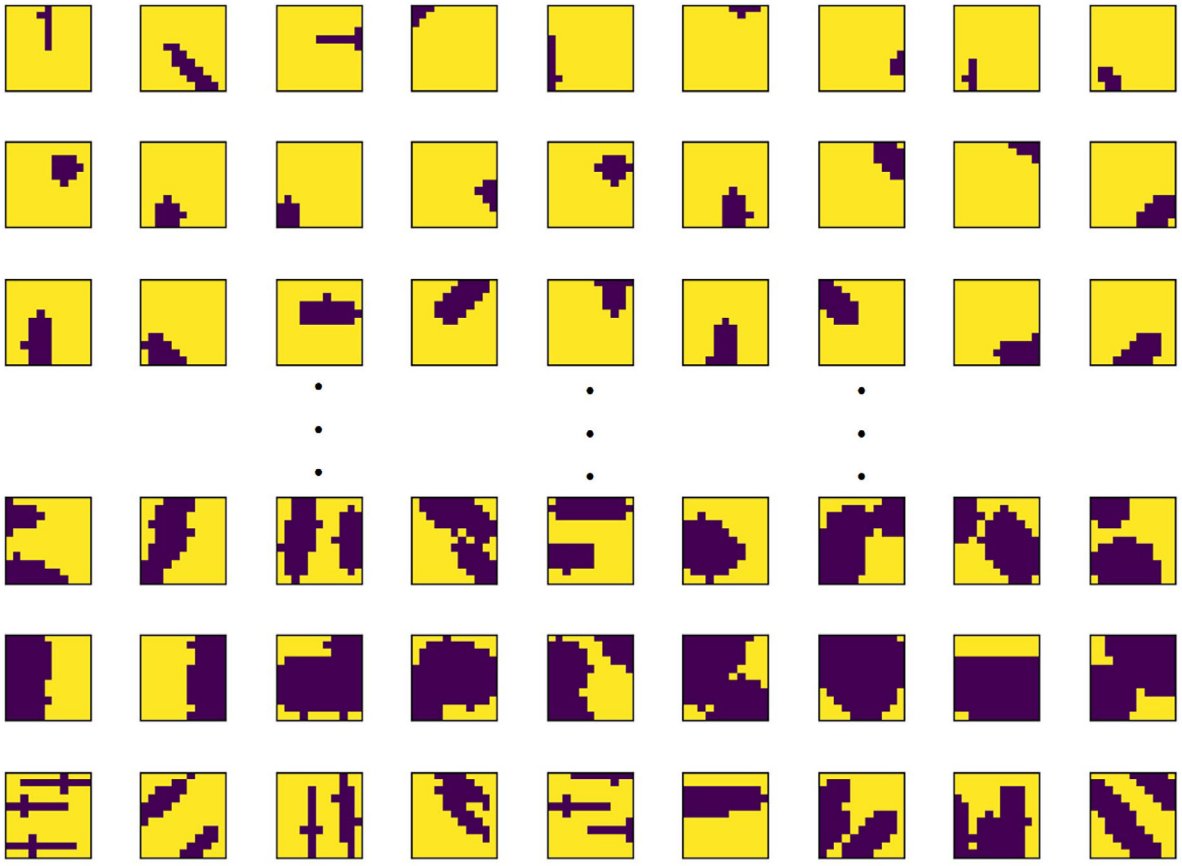


FIGURE B2 | Examples for random samples described in Sections 3.3 and 4.1. The samples are shown on 11×11 grid. Other resolutions are also available for training the MEA approach described in Section 4.5.

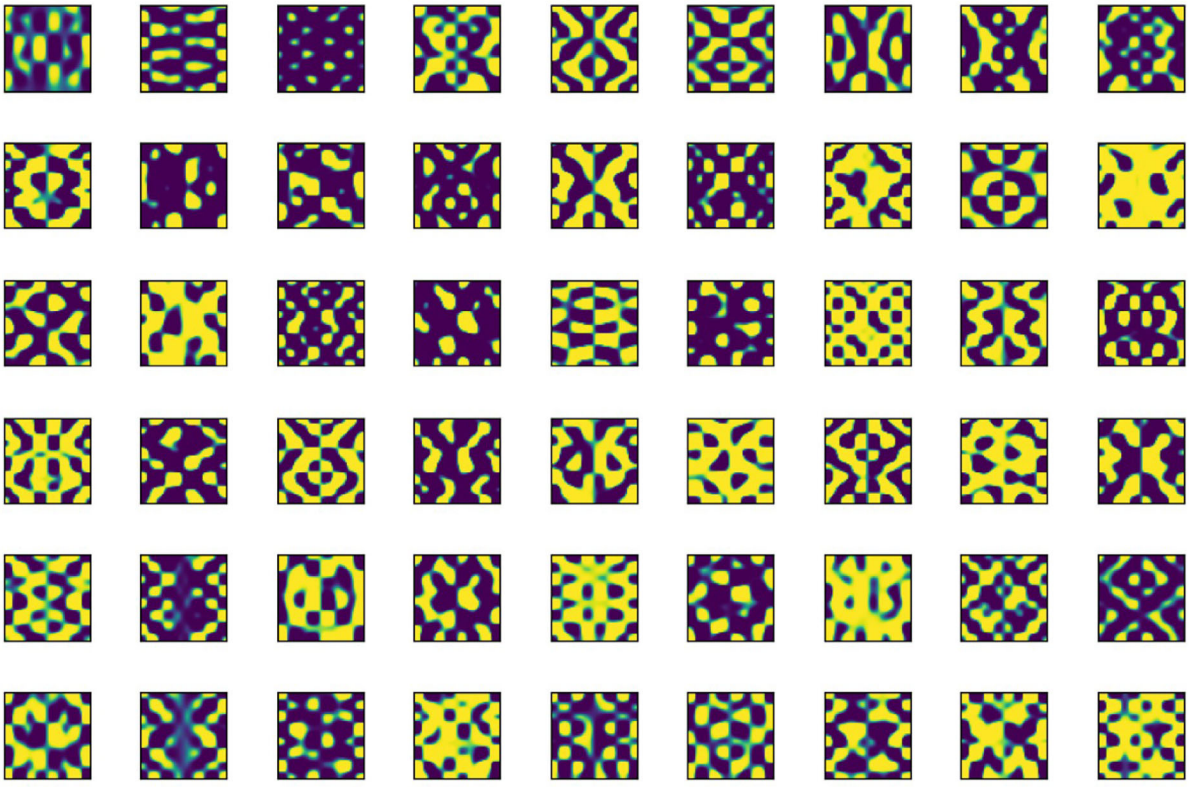


FIGURE B3 | Examples for random samples used for training of the NN described in Section 4.6. The samples are constructed on 51×51 grid.

10970207, 2025, 1, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/nme.7637 by Rwth Aachen University Universitatsbibliothek - 021000, Wiley Online Library on [09/01/2025]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License