Computers and Structures 263 (2022) 106742

Contents lists available at ScienceDirect

Computers and Structures

journal homepage: www.elsevier.com/locate/compstruc

Finite strain FE^2 analysis with data-driven homogenization using deep neural networks

Nan Feng, Guodong Zhang, Kapil Khandelwal *

Dept. of Civil & Env. Engg. & Earth Sci., 156 Fitzpatrick Hall, University of Notre Dame, Notre Dame, IN 46556, United States

ARTICLE INFO

Article history:
Received 21 September 2021
Accepted 7 January 2022
Available online 1 February 2022

Keywords:
FE^2 analysis
Finite strain homogenization
Deep neural networks
Data-driven methods
Hyperelasticity

ABSTRACT

A data-driven deep neural network (DNN) based approach is presented to accelerate FE^2 analysis.

It is computationally expensive to perform multiscale FE^2 analysis since at each macroscopic integration point an independent microscopic finite element analysis is needed. To alleviate this computational burden, DNN based surrogates are proposed for nonlinear homogenization that can serve as effective macroscale material models. A probabilistic approach is considered for surrogates' development, and an efficient data sampling strategy from the macroscopic deformation space is used for generating training and validation datasets. Frame indifference of macroscopic material behavior is consistently handled, and two training methods – regular training where only input/output pairs are included in the training dataset via L^2 loss function, and Sobolev training where the derivative data is also used with the Sobolev loss function – are compared. Numerical results demonstrate that Sobolev training leads to a higher testing accuracy as compared to regular training, and DNNs can serve as efficient and accurate surrogates for nonlinear homogenization in computationally expensive multiscale FE^2 analysis.

1. Introduction

In recent years, deep neural network (DNN) based modeling and simulation approaches have received considerable attention due to the rapid development in computational hardware (e.g., high-performance GPU and CPU devices) and availability of big data [1–3]. Historically, after the development of back-propagation algorithms in 1986 [4], researchers have successfully trained neural networks for different applications. In machine learning, back-propagation optimization algorithms have been used for image recognition tasks with convolutional neural networks (CNN) [5–7], and in regression problems with long sequences with long short-term memory (LSTM) recurrent neural networks (RNN) [8–10], among others. However, it was not until 2012 when the first deep neural network called Alexnet was successfully trained for computer vision applications, did the deep neural networks start receiving unprecedented attention. Alexnet used ReLU activations to increase the training efficiency and was trained using multiple GPUs [11]. Since then more efforts have been made towards designing efficient DNN architectures (e.g., GoogleNet [3], ResNet [12], and other densely connected networks [13]) and training/learning algorithms. For example, advanced weight initialization techniques were proposed to address gradient vanishing/exploding issues when training deep neural networks [14–17]; batch normalization layer was developed to avoid internal covariate drift issues [18]; regularization techniques like dropout were designed to address the overfitting issues [19]; and optimization algorithms based on mini-batch stochastic gradient methods such as Adam optimizer [20] were developed to increase efficiency in training deep networks. With these developments, DNN models are now widely used in applications ranging from image classification [11,21], speech recognition [22], machine translation systems [23,24], to natural language processing [25,26], among many others.

With these advancements in DNNs, there is also an increasing interest in the use of DNN models in computational mechanics [27–32]. In particular, DNN can provide efficient surrogates in various tasks which can lead to increased computational efficiency. This is especially true for computational multiscale analysis where finite element analyses across multiple scales have to be carried out concurrently [33]. Multiscale analysis is employed in many applications [34,35] and is indispensable for heterogeneous materials where complex mechanical properties are needed to be captured, e.g., fibrous materials [36], contact modeling at microscale [37], fracture propagation at microscale [38], etc. Previous studies have explored different strategies for leveraging DNN to speed up computations in multiscale mechanics. For instance, Unger and

* Corresponding author.
E-mail address: kapil.khandelwal@nd.edu (K. Khandelwal).

https://doi.org/10.1016/j.compstruc.2022.106742
0045-7949/

Könke adopted multilayer perceptron neural networks to approximate the homogenized strain–stress relationship for concrete modeling, where softening due to damage is included [39]. The authors later used similar neural networks for replacing the macroscopic stress and crack opening response at the interface between concrete and reinforcement [40]. Le et al. [41] explored the use of neural networks to approximate the homogenized constitutive behavior of small-strain nonlinear elastic unit cells with random microstructures that are described by geometric and material descriptors (e.g., size of inclusion, phase material properties, etc.). As a result, the input of the surrogate includes both macro strain measure and geometric and material descriptors, while the output is the homogenized strain energy. The homogenized stress and tangent are then obtained through direct differentiation. This work was further extended to the finite strain regime by Nguyen-Thanh et al. [42]. In another study, Bessa et al. [43] explored both kriging and neural network models as surrogates for homogenization of RVE made of hyperelastic phases with random microstructures described through a set of geometric descriptors. The authors also explored model reduction methods to reduce computational expense in order to build large databases for expensive RVE evaluations, which include plasticity and damage for further data-driven tasks. Huang et al. [44] used feedforward neural network as surrogate for linear homogenization analysis where the training data consists of indirect data such as load/displacement samples rather than direct observations of strain/stress data. This work was later extended by the authors to hyperelasticity and small strain elastoplasticity [45], where the lower triangular matrix from the Cholesky decomposition of the tangent moduli, which is assumed to be positive-definite, is learned using neural networks. Focusing on elastic problems, Fernández et al. [46] examined three DNN models for learning the homogenized constitutive laws – one for material strain/stress relationship, one for material strain and strain energy mapping, and one for strain energy function in terms of the material strain invariants – of an elastic/hyperelastic cubic lattice with internal symmetries. Under small strain assumptions, Ghavamian et al. [47] explored the use of a recurrent neural network (RNN) for homogenization of elastoplastic solids in the FE^2 analysis with a sequence-to-sequence input (macro strain) and output (macro stress) relationship. Logarzo et al. [48] developed a RNN based surrogate, which given the sequence of macro strain (representing a loading path) outputs sequences of both homogenized stress and user-defined localized quantities of interest (e.g., maximum plastic strain in the RVE domain). Wu et al. [49] considered the use of RNN as a surrogate model for the homogenization of RVE consisting of elastoplastic phases under finite strains, where the strain sequence in each training sample is obtained through a sequence of random walk in the strain space. In another study, with underlying hyperelastic constituents, Linka et al. [50] constructed a machine learning constitutive model using a general anisotropic hyperelasticity theory where the strain energy is written as a function of generalized invariants. Compared to the aforementioned studies, where material and geometric descriptors serve as input for surrogate models, Rao and Liu [51] proposed a 3D convolutional neural network (CNN) based surrogate for linear homogenization, where for a given 3D microstructure image as input, the elasticity tensor is predicted. Xiao et al. [52] conducted both regression (for linear constitutive modeling using Gaussian process and support vector machines) and classification (for failure probability approximation using DNNs) tasks for homogenization analysis of metal-ceramic composites considering microstructural uncertainties such as the size, shape and number of inclusions.

Although advancements have been made in the previous studies towards the development of data-driven surrogate models for multiscale simulations, many critical aspects are not completely addressed and remain unresolved. For instance, DNN surrogates are not constructed using probabilistic frameworks, and therefore, many theoretical and practical issues related to learning DNNs are not handled consistently. This is important especially if such models are to be extended for uncertainty quantification tasks. More importantly, in most of the past studies, the range of applicability of the developed surrogates is not clearly presented. As these data-driven models are mostly performing a regression task, it is important to quantify the range in which such surrogates are applicable. In general, under a pure data-driven framework, the surrogate models' results cannot be extrapolated outside the training range. Hence, the range of applicability of a surrogate model, i.e., working data space for the model, is of paramount importance and should be explicitly specified.

The goal of this paper is to develop a data-driven approach for accelerating constitutive modeling of periodic heterogeneous hyperelastic media in FE^2 analysis. To this end, a consistent probabilistic framework is presented within which DNN surrogates for constitutive models of periodic media are constructed. An efficient strategy for data generation in the macro principal stretch space is also proposed, and with a clear specification of training data space, a detailed study on the required size of the training dataset is also carried out. Multiple state-of-the-art neural network architectures are examined in search of effective DNN surrogates. In addition, an advanced learning strategy based on Sobolev training [53] is adopted for an improved training performance, where the 1st-order derivatives are incorporated in the loss function along with the target output labels. The efficacy of the proposed strategies is demonstrated on the representative examples within the context of finite strain FE^2 analysis.

The rest of this paper is organized as follows: Section 2 introduces the FE^2 analysis for multiscale mechanics problems. Section 3 presents probabilistic learning approaches for DNN models used in this study. Section 4 explores the performance of different DNN architectures and learning approaches in two homogenization problems. In Section 5, the trained DNN models are employed as surrogates in FE^2 analysis of three different structures. The conclusions of this paper are given in Section 6.

2. FE^2 analysis

For general heterogeneous media, constructing an analytical constitutive law can be challenging, if not impossible. In this section, the focus is on media that have periodic microstructures such that an unambiguous representation of the unit cell can be found. Assuming that the length scale (l) of the microstructure is much smaller than the characteristic length (L) of the macroscale problem, i.e., $L \gg l$, the 1st-order computational homogenization can be used to characterize the macroscopic material behaviors [33]. As a result, homogenization analysis is carried out to replace the traditional constitutive model evaluation at each integration point of the macroscale problem. The name FE^2 is coined since finite element analysis must be carried out at both the global macroscale level and integration point level in a nested way.

2.1. Macroscale problem

Consider a continuum body $\bar{\mathcal{B}}$ that undergoes a motion that takes it from a reference configuration $\bar{\Omega}_0 \in \mathbb{R}^3$ to a spatial configuration $\bar{\Omega}_t$, where an overbar is used to denote terms at the macroscale. The boundary of $\bar{\Omega}_0$, denoted as $\partial\bar{\Omega}_0$, is decomposed into the disjoint sets $\partial\bar{\Omega}_{0u}$ and $\partial\bar{\Omega}_{0\sigma}$ such that $\partial\bar{\Omega}_0 = \partial\bar{\Omega}_{0u} \cup \partial\bar{\Omega}_{0\sigma}$ and $\partial\bar{\Omega}_{0u} \cap \partial\bar{\Omega}_{0\sigma} = \varnothing$ with prescribed motion on $\partial\bar{\Omega}_{0u}$ and traction on $\partial\bar{\Omega}_{0\sigma}$. Taking an arbitrary material point of the body with position vector $\bar{\boldsymbol{X}} \in \bar{\Omega}_0$ in the reference configura-

tion and $\bar{\boldsymbol{x}} \in \bar{\Omega}_t$ in the spatial configuration, the motion can be described by a smooth one-to-one mapping $\bar{\boldsymbol{x}} = \bar{\boldsymbol{\varphi}}(\bar{\boldsymbol{X}}, t)$ where $t \in \mathbb{R}^+$ denotes time (see Fig. 1). The associated macro deformation gradient is defined by $\bar{\boldsymbol{F}} = \nabla_{\bar{\boldsymbol{X}}} \bar{\boldsymbol{\varphi}}$ with $\det \bar{\boldsymbol{F}} > 0$ and $\nabla_{\bar{\boldsymbol{X}}}$ the material gradient operator, i.e., w.r.t. $\bar{\boldsymbol{X}}$. The macroscopic linear momentum balance under quasi-static conditions is described by the boundary value problem (BVP) that reads

$$\begin{cases} \nabla_{\bar{\boldsymbol{X}}} . \bar{\boldsymbol{P}} = \boldsymbol{0} \ \text{in} \ \bar{\Omega}_0 \\ \bar{\boldsymbol{u}} = \widehat{\bar{\boldsymbol{u}}} \ \text{on} \ \partial\bar{\Omega}_{0u} \\ \bar{\boldsymbol{P}} . \bar{\boldsymbol{N}} = \widehat{\bar{\boldsymbol{T}}} \ \text{on} \ \partial\bar{\Omega}_{0\sigma} \end{cases} \tag{1}$$

where $\bar{\boldsymbol{P}}$ is the 1st Piola-Kirchhoff (PK) stress tensor, and $\widehat{\bar{\boldsymbol{u}}}$ and $\widehat{\bar{\boldsymbol{T}}}$ are the prescribed displacement and 1st PK traction vectors, respectively, on the boundaries with unit outward normal $\bar{\boldsymbol{N}}$. Here Eq. $(1)_1$ represents the balance of linear momentum under the absence of body forces, while the balance of angular momentum is implicitly satisfied by the symmetry of Kirchhoff stress $\bar{\boldsymbol{\tau}}$, i.e., $\bar{\boldsymbol{P}} . \bar{\boldsymbol{F}}^T = \bar{\boldsymbol{F}} . \bar{\boldsymbol{P}}^T$.

With the prescribed boundary and loading conditions, the motion of the continuum body is then determined by the material's constitutive model. For example, for hyperelastic materials a free energy density function $\bar{\psi}(\bar{\boldsymbol{F}})$ is postulated such that the 1st PK stress is calculated as

$$\bar{\boldsymbol{P}} = \frac{\partial \bar{\psi}}{\partial \bar{\boldsymbol{F}}} \tag{2}$$

with the tangent moduli computed by

$$\bar{\mathbb{A}} = \frac{\partial \bar{\boldsymbol{P}}}{\partial \bar{\boldsymbol{F}}} = \frac{\partial^2 \bar{\psi}}{\partial \bar{\boldsymbol{F}} \partial \bar{\boldsymbol{F}}} \tag{3}$$

The implementation details on macroscale BVP are given in Appendix A.1. For general heterogeneous (hyperelastic) solids, constructing an analytical closed-form of the energy function $\bar{\psi}(\bar{\boldsymbol{F}})$ can be highly non-trivial [54,55]. To aid this process, within the FE^2 framework, the macroscopic constitutive laws are computationally obtained from the solution of a microscale homogenization analysis.

2.2. Microscale problem

Given a media with periodic microstructure, a unit cell can be found that can serve as a representative volume element (RVE). This RVE can then be used to characterize the macroscopic behavior of the media [56]. The RVE at a material point $\bar{\boldsymbol{X}}$ deforms from a reference configuration Ω_0 to the current configuration Ω_t through a smooth one-to-one mapping $\boldsymbol{\varphi}$, i.e., $\boldsymbol{x} = \boldsymbol{\varphi}(\boldsymbol{X}, t)$ with $\boldsymbol{X} \in \Omega_0$ and $\boldsymbol{x} \in \Omega_t$ the position vectors in the two configurations. The RVE domain consists of a solid part $\mathcal{B}_0$ and void part $\mathcal{H}_0$, i.e., $\Omega_0 = \mathcal{B}_0 \cup \mathcal{H}_0$, with boundaries $\partial\mathcal{B}_0 = \partial\Omega_0 \cup \partial\mathcal{H}_0$, see Fig. 1. It is noted that microscopic heterogeneities can also be introduced by incorporating multiple solid material phases at the microscale scale with or without a void phase. In this study, the microstructural stability is assumed to be preserved during the motion, such that the periodicity of the microstructure remains unchanged, and the deformed unit cell can still serve as the RVE for estimating the material homogenized properties. The theoretical and computational aspects of multiscale stability can be found in [57] and [56,58] and are out of the scope of this study.

In the deformation-driven 1st-order homogenization framework, the deformation of the microstructure located at $\bar{\boldsymbol{X}}$ is driven by a deformation gradient $\bar{\boldsymbol{F}}(\bar{\boldsymbol{X}}, t) = \nabla_{\bar{\boldsymbol{X}}} \bar{\boldsymbol{\varphi}} = \boldsymbol{I} + \nabla_{\bar{\boldsymbol{X}}} \bar{\boldsymbol{u}}$ where $\bar{\boldsymbol{u}} = \bar{\boldsymbol{x}} - \bar{\boldsymbol{X}}$ is the macroscopic displacement field. The macroscopic material properties (homogenized stress and tangent moduli) are then evaluated under this deformation mode. The microscopic displacement field $\boldsymbol{u}(\boldsymbol{X}, t)$ over the RVE domain Ω_0 is assumed to be driven by the macroscopic deformation $\bar{\boldsymbol{F}}(\bar{\boldsymbol{X}}, t)$ by

$$\boldsymbol{u}(\boldsymbol{X}, t; \bar{\boldsymbol{X}}) = (\bar{\boldsymbol{F}}(\bar{\boldsymbol{X}}, t) - \boldsymbol{I}) . \boldsymbol{X} + \tilde{\boldsymbol{u}}(\boldsymbol{X}, t; \bar{\boldsymbol{X}}) \tag{4}$$

where $\tilde{\boldsymbol{u}}(\boldsymbol{X}, t; \bar{\boldsymbol{X}})$ is the displacement fluctuation field. The corresponding microscopic deformation gradient is

$$\boldsymbol{F}(\boldsymbol{X}, t; \bar{\boldsymbol{X}}) = \bar{\boldsymbol{F}}(\bar{\boldsymbol{X}}, t) + \nabla_{\boldsymbol{X}} \tilde{\boldsymbol{u}}(\boldsymbol{X}, t; \bar{\boldsymbol{X}}) \tag{5}$$

where the gradient operator $\nabla_{\boldsymbol{X}}$ is w.r.t. microscale coordinates $\boldsymbol{X}$. In the following, the dependence of the fields $\boldsymbol{u}$, $\tilde{\boldsymbol{u}}$ and $\boldsymbol{F}$ on $\bar{\boldsymbol{X}}$ is not explicitly expressed for the notational convenience, e.g.,

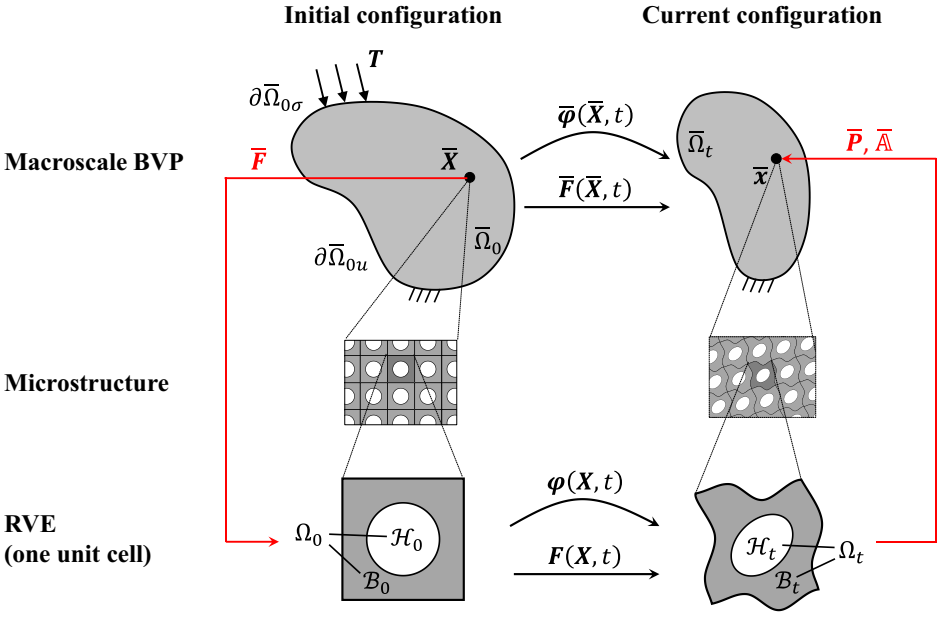


Fig. 1. Illustration of the FE^2 analysis of solids with periodic microstructures.

$\boldsymbol{u}(\boldsymbol{X},t)$ is used instead of $\boldsymbol{u}(\boldsymbol{X},t;\bar{\boldsymbol{X}})$. Following [59], the microscale displacement field has to satisfy the kinematical admissibility constraints

$$\int_{\mathcal{B}_0}\boldsymbol{u}(\boldsymbol{X},t)dV=\boldsymbol{0}\ \text{ and }\ \bar{\boldsymbol{F}}(\bar{\boldsymbol{X}},t)=\boldsymbol{I}+\frac{1}{V}\int_{\partial\Omega_0}\boldsymbol{u}(\boldsymbol{X},t)\otimes\boldsymbol{N}(\boldsymbol{X})dS \tag{6}$$

in which V is the volume of the domain Ω_0 and $\boldsymbol{N}$ is the unit normal vector on the boundary $\partial\Omega_0$. These kinematical admissibility constraints can be equivalently expressed in terms of the fluctuation field as

$$\int_{\mathcal{B}_0}\tilde{\boldsymbol{u}}(\boldsymbol{X},t)dV=\boldsymbol{0}\ \text{ and }\ \int_{\partial\Omega_0}\tilde{\boldsymbol{u}}(\boldsymbol{X},t)\otimes\boldsymbol{N}(\boldsymbol{X})dS=\boldsymbol{0} \tag{7}$$

where it is assumed that the coordinate system on the microscale is chosen such that $\int_{\mathcal{B}_0}\boldsymbol{X}dV=0$. It can be seen that the constraint in Eq. $(7)_1$ is equivalent to removing rigid-body translation, while the constraint in Eq. $(7)_2$ implicitly removes rigid-body rotation. The constraints in Eq. (6) or (7) correspond to the *constant traction boundary conditions* [56], i.e.,

$$\boldsymbol{T}(\boldsymbol{X},t)=\bar{\boldsymbol{P}}(\bar{\boldsymbol{X}},t).\boldsymbol{N}(\boldsymbol{X})\ \text{on}\ \partial\Omega_0 \tag{8}$$

where $\boldsymbol{T}(\boldsymbol{X},t)\overset{\text{def}}{=}\boldsymbol{P}(\boldsymbol{X},t).\boldsymbol{N}(\boldsymbol{X})$ represents the 1st PK (nominal) traction acting on the reference surface with normal $\boldsymbol{N}(\boldsymbol{X})$; $\boldsymbol{P}(\boldsymbol{X},t)$ denotes the microscopic 1st PK stress field corresponding to a material point $\bar{\boldsymbol{X}}$ at the macroscale, while $\bar{\boldsymbol{P}}(\bar{\boldsymbol{X}},t)$ represents the macroscopic/homogenized 1st PK stress (see Eq. (11)). Additional constraints can be introduced in a consistent way that may lead to periodic boundary conditions or linear displacement boundary conditions [56,59].

Since periodic microstructures are considered in this study, the periodic boundary conditions are used, where the boundary $\partial\Omega_0$ is divided into a pair of negative and positive sides, $\partial\Omega_0^+$ and $\partial\Omega_0^-$, periodicity implies

$$\tilde{\boldsymbol{u}}^+=\tilde{\boldsymbol{u}}^-\quad\text{on}\quad\partial\Omega_0 \tag{9}$$

where $\tilde{\boldsymbol{u}}^+$ and $\tilde{\boldsymbol{u}}^-$ are displacement fluctuations on the negative side and positive side, respectively. From periodicity, any point on the positive side can be reached by translating the corresponding point on the negative side with a periodic lattice vector $\boldsymbol{a}_1$ or $\boldsymbol{a}_2$ or $\pm(\boldsymbol{a}_1\pm\boldsymbol{a}_2)$, see Fig. 2. The boundary constraint in Eq. (9) automatically satisfies the constraints in Eqs. (6) or (7).

The transition between the micro and macro scales is governed by the Hill-Mandel condition [60,61], which states that $\forall\bar{\boldsymbol{X}}\in\bar{\Omega}_0$, $t\in\mathbb{R}^+$ and kinematically admissible $\tilde{\boldsymbol{u}}$

$$\bar{\boldsymbol{P}}:\delta\bar{\boldsymbol{F}}=\frac{1}{V}\int_{\mathcal{B}_0}\boldsymbol{P}:\delta\boldsymbol{F}dV \tag{10}$$

which implies the equivalence of the incremental virtual work between the micro and macro scales.

The stress homogenization relation

$$\begin{aligned}\bar{\boldsymbol{P}}&=\frac{1}{V}\int_{\mathcal{B}_0}\boldsymbol{P}dV=\frac{1}{V}\int_{\partial\mathcal{B}_0}\boldsymbol{T}\otimes\boldsymbol{X}dS\\&\equiv\frac{1}{V}\int_{\partial\Omega_0}\boldsymbol{T}\otimes\boldsymbol{X}dS\quad\text{with}\ \boldsymbol{T}=\boldsymbol{P}.\boldsymbol{N}\end{aligned} \tag{11}$$

and the weak form of the microscale equilibrium equation

$$\int_{\mathcal{B}_0}\boldsymbol{P}:\nabla_{\boldsymbol{X}}\delta\tilde{\boldsymbol{u}}\,dV=\boldsymbol{0} \tag{12}$$

can be obtained from Eq. (10) by choosing $\delta\tilde{\boldsymbol{u}}=\boldsymbol{0}$ and $\delta\bar{\boldsymbol{F}}=\boldsymbol{0}$, respectively. Here, the second equality in Eq. (11) can be proved using the divergence theorem and the fact that $\nabla_{\boldsymbol{X}}.\boldsymbol{P}=\boldsymbol{0}$, while the third equality is due to the traction-free void boundaries, i.e., $\boldsymbol{T}=\boldsymbol{0}$ on $\partial\mathcal{H}_0$. The implementation details on the homogenization analysis are given in Appendix A.2.

3. Data-driven nonlinear homogenization

Although the FE^2 framework presented in Section 2 accommodates inelastic constitutive behaviors, the focus of this study is on finite strain hyperelastic solids. As a result, the constitutive laws of the microstructure constituents are path independent. However, FE^2 analysis can still be computationally prohibitive as it requires an independent nonlinear finite element analysis at each integration point of the macroscale BVP. To relieve the computational burden, a data-driven approach is proposed in this section to replace the underlying homogenization analysis at each integration point.

3.1. Input/output of the surrogate for homogenization

For homogenization analysis, the inputs include the RVE model (geometric microstructure and constituents' constitutive laws) and macroscopic deformation gradient $\bar{\boldsymbol{F}}$, while the outputs are the homogenized 1st PK stress $\bar{\boldsymbol{P}}$ and tangent moduli $\bar{\mathbb{A}}$ $(=\partial\bar{\boldsymbol{P}}/\partial\bar{\boldsymbol{F}})$. In this study, the data-driven surrogate models are developed for RVEs with prescribed geometric microstructures and constitutive laws. Hence, developed surrogate models are intended to work for the specified RVE model and cannot be used for arbitrary (stochastic) RVEs. Therefore, the input and output pairs for the surrogate model for a given RVE are $\bar{\boldsymbol{F}}$ and $\bar{\boldsymbol{P}}$, respectively. In other words, the mapping defined from the homogenization of a given RVE, i.e., $\bar{\mathcal{M}}:\bar{\boldsymbol{F}}\to\bar{\boldsymbol{P}}$, is to be replaced by a DNN surrogate.

It is noted that the frame indifference requires that the mapping $\bar{\mathcal{M}}:\bar{\boldsymbol{F}}\to\bar{\boldsymbol{P}}$ should satisfy $\bar{\mathcal{Q}}.\bar{\mathcal{M}}(\bar{\boldsymbol{F}})=\bar{\mathcal{M}}(\bar{\mathcal{Q}}.\bar{\boldsymbol{F}})$ for any rotation tensor

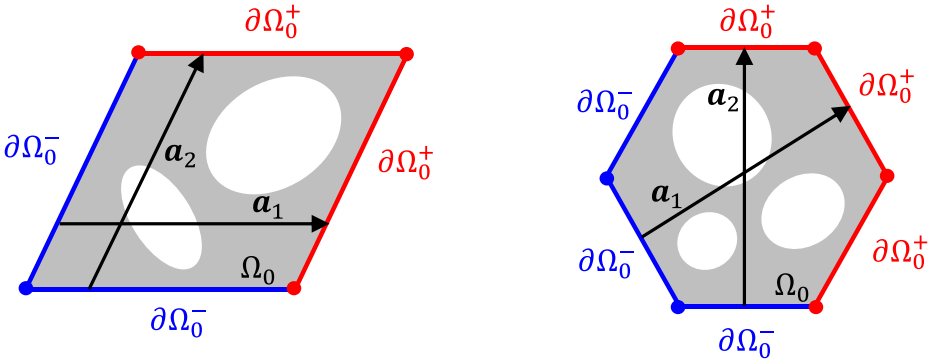


Fig. 2. Partitioning of boundary of RVE into positive and negative parts.

$\bar{\boldsymbol{Q}} \in \mathrm{SO}(3)$ [62]. Although this physics constraint can be enforced by adding penalization terms during the surrogate training phase or can be directly learned from data by increasing the data size, the training efficiency will deteriorate. To address this issue, the material tensor pair $(\bar{\boldsymbol{C}}, \bar{\boldsymbol{S}})$ is adopted as the input/output pair of the surrogate model, where $\bar{\boldsymbol{C}} = \bar{\boldsymbol{F}}^T.\bar{\boldsymbol{F}}$ is the macroscopic right Cauchy Green strain tensor and $\bar{\boldsymbol{S}}$ is the macroscopic 2nd PK stress. Using this input/output pair the frame indifference is automatically satisfied, as $\bar{\boldsymbol{C}}$ and $\bar{\boldsymbol{S}}$ are frame invariant. Therefore, the surrogates are developed to learn the mapping $\mathbb{M} : \bar{\boldsymbol{C}} \rightarrow \bar{\boldsymbol{S}}$ from the data. The homogenized 1st PK stress $\bar{\boldsymbol{P}}$ and tangent moduli $\bar{\mathbb{A}}$ that are needed in FE^2 are recovered by the push-forward operations

$$\bar{\boldsymbol{P}} = \bar{\boldsymbol{F}}.\bar{\boldsymbol{S}} \quad \text{and} \quad \bar{\mathbb{A}}_{ijkl} = F_{ip}F_{kq}\bar{\mathbb{C}}_{pjql} \quad \text{with} \quad \bar{\mathbb{C}} \stackrel{\text{def}}{=} 2\frac{\partial \bar{\boldsymbol{S}}}{\partial \bar{\boldsymbol{C}}} \tag{13}$$

Hence, this problem can be categorized as a regression problem, where the target mapping $\bar{\boldsymbol{C}} \overset{\mathbb{M}}{\mapsto} \bar{\boldsymbol{S}}$ is learned from the data. Furthermore, to enforce the symmetry of $\bar{\boldsymbol{S}}$, only the symmetric parts of $\bar{\boldsymbol{C}}$ and $\bar{\boldsymbol{S}}$ are used, e.g., the size of the input or output tensor is 6×1 for 3D while 3×1 for 2D problems. The tangent moduli $\bar{\mathbb{C}}$ is obtained by direct differentiation of the surrogate model $\mathbb{M}$, i.e.,

$$\bar{\mathbb{C}} = 2\frac{\partial \mathbb{M}}{\partial \bar{\boldsymbol{C}}} \tag{14}$$

The flowchart of FE^2 analysis with the direct homogenization analysis or surrogate evaluation is shown in Fig. 3. Note that the homogenization analysis requires a nonlinear finite element analysis of the unit cell at each macroscale integration point and the main computational expense in FE^2 is associated with this nested analysis step. However, the use of surrogate models will obviate this expensive analysis step that is required at a macroscale integration point. In this sense, the surrogate model $\mathbb{M}$ directly provides the derivatives of the homogenized hyperelastic free energy for the periodic material. The following subsections present two different unit cells microstructures that are considered for FE^2 analysis in this paper.

3.1.1. Unit cell-1: three-phase periodic solids

In the first case, a unit cell with three material phases is considered as shown in Fig. 4. The three material phases are all hyperelastic materials but with different stiffness that is modeled by a neo-Hookean constitutive model (see Eq. (A9)). The geometry, FE mesh, and model parameters of the unit cell are also shown in Fig. 4. For this unit cell, standard 4-node quadrilateral elements are used to discretize the domain.

3.1.2. Unit cell-2: two-phase periodic solids

The second case considers a two-phase unit cell with a leaf-shaped inclusion. Again, the underlying materials follow the neo-Hookean model with parameters given in Fig. 5, where the geometry and FE mesh information is also provided. In this unit cell, due to the geometric complexity of the domain, linear triangle elements are used for the FE discretization.

3.2. Probabilistic background of the regression problem

Consider a dataset of random variables $\mathcal{D}_{\text{train}} = \left\{\boldsymbol{Z}^{(r)} := \boldsymbol{Y}^{(r)}|\boldsymbol{X}^{(r)}\right\}_{r=1}^{m}$ of m i.i.d samples, i.e., $\boldsymbol{Z}^{(r)} \sim p\left(\boldsymbol{z}^{(r)}\right) = p\left(\boldsymbol{y}^{(r)}|\boldsymbol{x}^{(r)}\right)$, where $\boldsymbol{X}^{(r)} \in \mathbb{R}^d$ and $\boldsymbol{Y}^{(r)} \in \mathbb{R}^d$, a random variable $\boldsymbol{Z} = \boldsymbol{Y}|\boldsymbol{X} \sim p(\boldsymbol{y}|\boldsymbol{x})$, and the random model parameters $\boldsymbol{\Theta} \sim p(\boldsymbol{\theta})$ that are assumed to respect the graph in Fig. 6. The joint probability density function (PDF), $p(\boldsymbol{z}, D_{\text{train}}, \boldsymbol{\theta})$, can be then expressed as

$$p(\boldsymbol{z}, D_{train}, \boldsymbol{\theta}) = p(\boldsymbol{\theta})p(D_{train}|\boldsymbol{\theta})p(\boldsymbol{z}|\boldsymbol{\theta}) \tag{15}$$

where

$$p(D_{\text{train}}|\boldsymbol{\theta}) = \prod_{r=1}^{m} p\left(\boldsymbol{z}^{(r)}|\boldsymbol{\theta}\right) \tag{16}$$

In Eq. (15), $p(\boldsymbol{\theta})$ is the prior PDF of the parameters Θ, $p(\boldsymbol{z}|\boldsymbol{\theta})$ is the conditional PDF of data, and $p(D_{\text{train}}|\boldsymbol{\theta})$ is the conditional probability of the data. In a regression task, the goal is to infer the conditional PDF $p(\boldsymbol{z}|D_{\text{train}})$, which is also known as *posterior predictive* PDF, for a given prior PDF $(p(\boldsymbol{\theta}))$ and the conditional PDF of the data $(p(\boldsymbol{z}|\boldsymbol{\theta}))$. This posterior predictive PDF is given by

$$p(\boldsymbol{z}|D_{\text{train}}) = \frac{\mathbb{E}_{\boldsymbol{\Theta}\sim\, p(\boldsymbol{\theta})}[p(D_{\text{train}}|\Theta)p(\boldsymbol{z}|\Theta)]}{\mathbb{E}_{\boldsymbol{\Theta}\sim\, p(\boldsymbol{\theta})}[p(D_{\text{train}}|\Theta)]} \tag{17}$$

Further simplifications are made in probabilistic point estimation approaches, e.g., maximum likelihood estimation (MLE) and maximum a posteriori (MAP) estimation [63], wherein for the calculation of the *posterior predictive* PDF $p(\boldsymbol{z}|D_{\text{train}})$, it is assumed that the model parameters Θ have negligible uncertainties, i.e., $p(\boldsymbol{\theta}) = \delta\left(\boldsymbol{\theta} - \boldsymbol{\theta}\right)$ where $\boldsymbol{\theta} = \mathbb{E}_{\boldsymbol{\Theta}\sim\, p(\boldsymbol{\theta})}[\boldsymbol{\Theta}]$ are the unknown parameters. With this assumption, the posterior predictive PDF $p(\boldsymbol{z}|D_{\text{train}})$ is obtained as

$$p(\boldsymbol{z}|D_{\text{train}}) = \frac{p\left(D_{\text{train}}|\,\boldsymbol{\theta}\right)p\left(\boldsymbol{z}|\,\boldsymbol{\theta}\right)}{p\left(D_{\text{train}}|\,\boldsymbol{\theta}\right)} = p\left(\boldsymbol{z}|\,\boldsymbol{\theta}\right) \tag{18}$$

Thus, in the case of point estimation methods, the goal is to determine the model parameters $\boldsymbol{\theta}$ that can be used to describe the conditional data density $p\left(\boldsymbol{z}|\,\boldsymbol{\theta}\right)$. In regression, a common model for the conditional data density $p\left(\boldsymbol{z}|\,\boldsymbol{\theta}\right)$ is the Gaussian PDF [63], i.e.,

$$p\left(\boldsymbol{z}|\widetilde{\boldsymbol{\theta}}\right) = \mathcal{N}(\boldsymbol{y}|\mathcal{F}(\boldsymbol{x};\boldsymbol{w}), \boldsymbol{\Sigma}) \quad \text{with} \quad \boldsymbol{\Sigma} = \sigma^2\boldsymbol{I} \text{ and } \widetilde{\boldsymbol{\theta}} \stackrel{\text{def}}{=} \{\boldsymbol{w}, \sigma^2\} \tag{19}$$

where the function $\mathcal{F}(x;w)$ with unknown parameters, $\boldsymbol{w}$, specifies the mean of the conditional PDF $p\left(\boldsymbol{z}|\,\boldsymbol{\theta}\right)$. In a regression task, the parametric form of the mean function $\mathcal{F}(x;w)$ has to be specified in terms of parameters $\boldsymbol{w}$ and many functional forms of this dependence can be considered [63]. In particular, in deep machine learning, neural networks are used as mean function approximators to express this parametric dependence, and the learning task is then to estimate the model parameters $\boldsymbol{\theta}$ by $\widehat{\boldsymbol{\theta}} = \{\widehat{\boldsymbol{w}}, \widehat{\sigma}^2\}$ from the available data D_{train}. After the model parameters are estimated, the deterministic regression is described in terms of the most probable estimate of $\widehat{\boldsymbol{y}}$ of $\boldsymbol{Y}$ given $\boldsymbol{X} = \boldsymbol{x}$ as

$$\widehat{y} = \mathcal{F}(x;\widehat{w}) \tag{20}$$

For the FE^2 surrogates developed in this paper, the input $\boldsymbol{x}$ is taken as the macroscopic right Cauchy Green strain tensor $\bar{\boldsymbol{C}}$, the output $\widehat{\boldsymbol{y}}$ is taken as the macroscopic 2nd-Piola Kirchhoff stress tensor $\bar{\boldsymbol{S}}$ and the learned mapping $\mathcal{F} \equiv \mathcal{M}$.

3.3. Regular training

In the case of maximum likelihood estimation (MLE), the model parameters $\boldsymbol{\theta}$ in Eq. (19) are determined by maximizing the probability of data which is expressed as

$$\mathcal{L}\left(\boldsymbol{\theta}\right) \stackrel{\text{def}}{=} p\left(D_{\text{train}}|\,\boldsymbol{\theta}\right) = \prod_{r=1}^{m} p\left(\boldsymbol{y}^{(r)}|\boldsymbol{x}^{(r)}, \boldsymbol{\theta}\right) \tag{21}$$

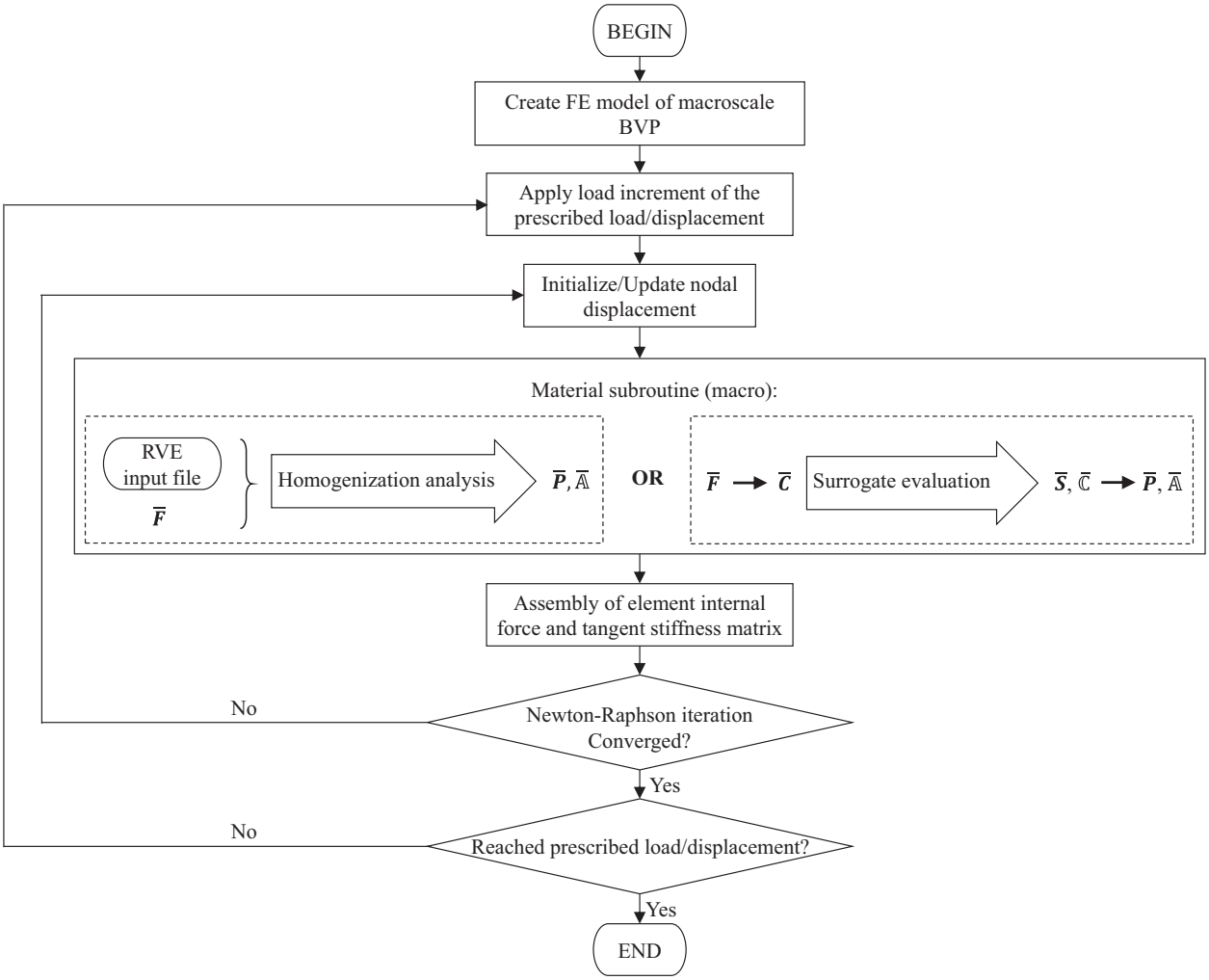


Fig. 3. Flowchart of FE^2 analysis with direct homogenization analysis or surrogate (DNN).

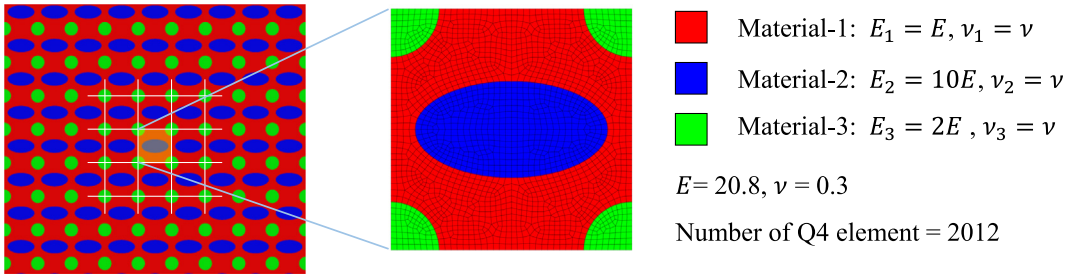


Fig. 4. Geometry, FE mesh, and model parameters of the unit cell-1.

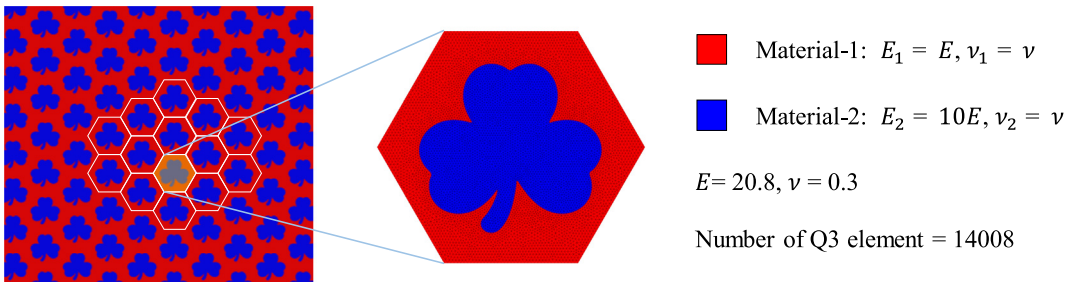


Fig. 5. Geometry, FE mesh, and model parameters of the unit cell-2.

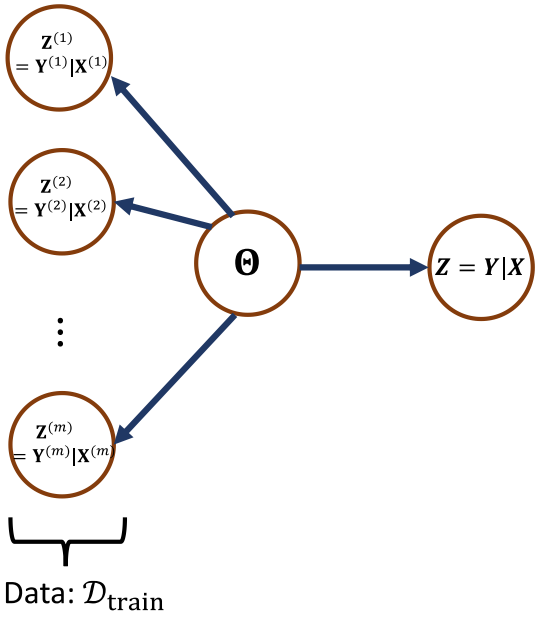


Fig. 6. Probabilistic graphical model for regression.

where $\mathcal{L}\left(\boldsymbol{\theta}\right)$ is the likelihood function for the parametric family $p\left(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta}\right)$. In practice, to avoid the issues related to numerical underflow, the log-likelihood function is considered, i.e.,

$$\ell\left(\boldsymbol{\theta}\right) \overset{\text{def}}{=} \ln \mathcal{L}\left(\boldsymbol{\theta}\right) = \sum_{r=1}^{m} \ln p\left(\boldsymbol{y}^{(r)}|\boldsymbol{x}^{(r)},\boldsymbol{\theta}\right) \tag{22}$$

Finally, MLE of the unknown parameters $\boldsymbol{\theta}$ is obtained by maximizing $\ell\left(\boldsymbol{\theta}\right)$, i.e.

$$\widehat{\boldsymbol{\theta}}_{MLE} \in \underset{\boldsymbol{\theta}}{\operatorname{argmin}} -\ell\left(\boldsymbol{\theta}\right) \tag{23}$$

with

$$\begin{aligned}\ell\left(\widetilde{\boldsymbol{\theta}}\right) &= \sum_{r=1}^{m} \ln p\left(\boldsymbol{y}^{(r)}\middle|\boldsymbol{x}^{(r)},\widetilde{\boldsymbol{\theta}}\right) \\ &= -\frac{1}{2\sigma^2}\sum_{r=1}^{m}\left\|\boldsymbol{\mathcal{F}}\left(\boldsymbol{x}^{(r)};\boldsymbol{w}\right) - \boldsymbol{y}^{(r)}\right\|^2 - \frac{md}{2}\ln(2\pi\sigma^2)\end{aligned} \tag{24}$$

where the Gaussian PDF of $p\left(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta}\right)$ in Eq. (19) is used. By considering the stationary conditions $\partial\ell\left(\boldsymbol{\theta}\right)/\partial\boldsymbol{w} = 0$ and $\partial\ell\left(\boldsymbol{\theta}\right)/\partial\sigma^2 = 0$, the MLE of $\boldsymbol{w}$, i.e., $\widehat{\boldsymbol{w}}_{MLE}$, is obtained by solving a least squares optimization problem

$$\widehat{\boldsymbol{w}}_{MLE} \in \underset{\boldsymbol{w}}{\operatorname{argmin}} L_{MSE}(\boldsymbol{w}) \tag{25}$$

$$L_{MSE}(\boldsymbol{w}) \overset{\text{def}}{=} \frac{1}{m}\sum_{r=1}^{m}\left\|\boldsymbol{\mathcal{F}}\left(\boldsymbol{x}^{(r)};\boldsymbol{w}\right) - \boldsymbol{y}^{(r)}\right\|^2 \tag{26}$$

and $\widehat{\sigma}^2_{MLE}$ is computed by

$$\widehat{\sigma}^2_{MLE} = \frac{1}{d} L_{MSE}\left(\widehat{\boldsymbol{w}}_{MLE}\right) \tag{27}$$

In general, the MSE loss function $L_{MSE}(\boldsymbol{w})$ in Eq. (26) is nonconvex and this optimization problem can be solved using nonlinear programming methods, e.g., nonlinear least squares [64]. In machine learning, where neural networks are used to parametrize the function $\mathcal{F}(x;w)$ and the size of the dataset (m) is large, this optimization problem is then usually solved using stochastic optimization approaches (SOA) [20]. To this end, the loss function $L_{MSE}(\boldsymbol{w})$ is reformulated in an *expectation form*, i.e.,

$$L_{MSE}(\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{D}\sim\, p_{data}(\boldsymbol{d})}\left[\left\|\bar{L}(\boldsymbol{D})\right\|^2\right] \text{ with } \bar{L}(\boldsymbol{d}) \overset{\text{def}}{=} \mathcal{F}(x;w) - y$$

$$p_{data}(\boldsymbol{d}) = \frac{1}{m}\sum_{r=1}^{m}\delta\left(\boldsymbol{d} - \boldsymbol{d}^{(r)}\right) \tag{28}$$

where $\boldsymbol{d}^{(r)} = \left(\boldsymbol{x}^{(r)},\boldsymbol{y}^{(r)}\right)$ is the r th data point in the training dataset and $p_{data}(\boldsymbol{d})$ is a discrete empirical data distribution defined by the training data. With the MSE loss function expressed in the expectation form shown in Eq. (28), the SOA can be used to solve this problem [20]. The main advantage of using SOA is that the optimization can proceed by using only a small subset of the datasets, i.e., by using mini-batches, which makes the optimization process more efficient, especially when large datasets are used [65].

3.4. Sobolev training – With 1st-order derivative data

Recently, the idea of Sobolev training for neural networks is introduced in [53], where the derivatives of mapping were also used to learn the underlying mapping from the data. It was shown in [53] that incorporating the derivatives information in neural network training improves not only the accuracy of the network prediction but also the data efficiency and generalization capabilities of the learned mapping. More specifically, consider a dataset $D_{\text{train}} = \left\{\left(\boldsymbol{x}^{(r)},\boldsymbol{y}^{(r)},\boldsymbol{J}^{(r)}\right)\right\}_{r=1}^{m}$ where $\boldsymbol{J}^{(r)} \overset{\text{def}}{=} \left.\dfrac{\partial\widehat{\mathcal{F}}}{\partial\boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}^{(r)}}$ and $\widehat{\mathcal{F}}: x \to y$ is a smooth mapping between inputs $\boldsymbol{x}^{(r)}$ and outputs $\boldsymbol{y}^{(r)}$ that generates the data. Using Sobolev training [53], an optimal estimate of $\boldsymbol{w}$ is then found by optimizing a multi-objective function $L_c(\boldsymbol{w})$, i.e.,

$$\widehat{\boldsymbol{w}} \in \underset{\boldsymbol{w}}{\operatorname{argmin}} L_c(\boldsymbol{w}) \tag{29}$$

$$\begin{aligned}L_c(\boldsymbol{w}) \overset{\text{def}}{=}\ & \frac{1}{m}\sum_{r=1}^{m}\left\|\boldsymbol{\mathcal{F}}\left(\boldsymbol{x}^{(r)};\boldsymbol{w}\right) - \boldsymbol{y}^{(r)}\right\|^2 \\ & + \lambda_p\left(\sum_{r=1}^{m}\left\|\left.\frac{\partial\boldsymbol{\mathcal{F}}(\boldsymbol{x};\boldsymbol{w})}{\partial\boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}^{(r)}} - \boldsymbol{J}^{(r)}\right\|^2\right)\end{aligned} \tag{30}$$

where $\lambda_p > 0$ is a user-defined hyperparameter that serves as a weighting factor between the two loss terms. For using SOA, the objective function $L_c(\boldsymbol{w})$ is again expressed in an expectation form as

$$L_c(\boldsymbol{w}) = \mathbb{E}_{\boldsymbol{U}\sim\, p_{data}(\boldsymbol{u})}\left[\left\|\bar{L}(\boldsymbol{U})\right\|^2 + \lambda_p\left\|D(\boldsymbol{U})\right\|^2\right] \tag{31}$$

with

$$\begin{aligned}\bar{L}(\boldsymbol{u}) &\overset{\text{def}}{=} \boldsymbol{\mathcal{F}}(\boldsymbol{x};\boldsymbol{w}) - \boldsymbol{y} \quad \text{and} \quad D(\boldsymbol{u}) \\ &\overset{\text{def}}{=} \left.\frac{\partial\boldsymbol{\mathcal{F}}(\boldsymbol{x};\boldsymbol{w})}{\partial\boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}^{(r)}} - \boldsymbol{J}^{(r)}\end{aligned} \tag{32}$$

and

$$p_{data}(\boldsymbol{u}) = \frac{1}{m}\sum_{r=1}^{m}\delta\left(\boldsymbol{u} - \boldsymbol{u}^{(r)}\right) \tag{33}$$

where $\boldsymbol{u}^{(r)} = \left(\boldsymbol{x}^{(r)}, \boldsymbol{y}^{(r)}, \boldsymbol{J}^{(r)}\right)$ is the r^{th} data point in the training dataset.

Remark: From a probabilistic viewpoint, the solution of the optimization problem in Eq. (29) can be seen as a maximum a posteriori estimate (MAP) $\widehat{\boldsymbol{w}}_{MAP}$ of $\boldsymbol{w}$, in a sense that the prior distribution $p(\boldsymbol{w})$ of the parameters $\boldsymbol{w}$ has the following property

$$p(\boldsymbol{w}) = \frac{1}{C(\lambda_p)} \exp\left(-\lambda_p \sum_{r=1}^{m} \left[\left\| \left.\frac{\partial \mathcal{F}(\boldsymbol{x};\boldsymbol{w})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}^{(r)}} - \boldsymbol{J}^{(r)} \right\|^2\right]\right) \text{ with}$$

$$C(\lambda_p) = \int \exp\left(-\lambda_p \sum_{r=1}^{m} \left\| \left.\frac{\partial \mathcal{F}(\boldsymbol{x};\boldsymbol{w})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}^{(r)}} - \boldsymbol{J}^{(r)} \right\|^2\right) d\boldsymbol{w} \tag{34}$$

which means that the prior PDF belongs to one-parameter exponential family distribution [66] and can be rewritten as

$$p(\boldsymbol{w}) = h(\boldsymbol{w}) \exp\left(\eta(\lambda_p) T(\boldsymbol{w}) - A(\lambda_p)\right) \text{ with}$$

$$h(\boldsymbol{w}) = 1, A(\lambda_p) = \ln C(\lambda_p), \eta(\lambda_p) = -\lambda_p \text{ and}$$

$$T(\boldsymbol{w}) = \sum_{r=1}^{m} \left\| \left.\frac{\partial \mathcal{F}(\boldsymbol{x};\boldsymbol{w})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}^{(r)}} - \boldsymbol{J}^{(r)} \right\|^2 \tag{35}$$

where $T(\boldsymbol{w})$ can be shown to be the sufficient statistics [67], in the sense that by letting $V = T(\boldsymbol{W})$, the conditional PDF $p(\boldsymbol{w}|v)$ reads

$$p(\boldsymbol{w}|v) = \frac{p(\boldsymbol{w}, v)}{p(v)} = \frac{p(v|\boldsymbol{w})p(\boldsymbol{w})}{\mathbb{E}_{\boldsymbol{W}\sim\, p(\boldsymbol{w})}[p(v|\boldsymbol{W})]}$$

$$= \frac{p(v|\boldsymbol{w}) \exp(\eta v - A)}{\exp(\eta v - A) \int_{\Omega_{\boldsymbol{w}}} p(v|\boldsymbol{w}) d\boldsymbol{w}} = \frac{p(v|\boldsymbol{w})}{\int_{\Omega_{\boldsymbol{w}}} p(v|\boldsymbol{w}) d\boldsymbol{w}}$$

$$\text{with } p(v|\boldsymbol{w}) = \delta(v - T(\boldsymbol{w})) \text{ and } \Omega_{\boldsymbol{w}} = \{\boldsymbol{w} : T(\boldsymbol{w}) = v\} \tag{36}$$

and does not depend on λ_p. Hence, the optimization problem in Eq. (29) can be equivalently considered as optimizing the negative logarithm of the posterior PDF $p(\boldsymbol{w}|D_{\text{train}}) \propto p(\boldsymbol{w})p(D_{\text{train}}|\boldsymbol{w})$. From this viewpoint, incorporating the derivative data has a regularizing influence on the model, which leads to the aforementioned desirable features.

3.5. Stochastic optimization algorithm (SOA)

To solve the optimization problem in Eq. (25) or (29), mini-batch stochastic gradient descent is used. The main idea is to divide the training dataset D_{train} into n_b mini-batches, i.e. D_1, D_2, $\cdots$, D_{n_b}. The number of samples in each batch is $s_b = m/n_b$ and is called the *batch size*. Then, the gradient is evaluated over a mini-batch instead of the entire training dataset, and this gradient is used to update the model parameters. All mini-batch datasets are used one by one to compute a stochastic gradient as an approximation for the true gradient, and one cycle over all the mini-batches is termed as an *epoch*. Hence, one epoch consists of n_b optimization iterations. At the k th iteration within an epoch ($k = 1, 2, \cdots, n_b$), the stochastic gradient is given by $\left.\frac{\partial L_{D_k}(\boldsymbol{w})}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}=\boldsymbol{w}_k}$ where $L_{D_k}(\boldsymbol{w})$ denotes the loss function in Eq. (26) for regular training or Eq. (30) for Sobolev training evaluated by the mini-batch training data D_k. The evaluated stochastic gradient is, however, usually noisy and can be far away from true gradient, which can result in slow convergence. To address this issue, Adam optimizer [20] that considers moving statistics of the stochastic gradient is used to make an approximation for the true gradient. In the Adam optimization algorithm, the learning rate is adjusted at each iteration. At k^{th} optimization step in each epoch, the update of an arbitrary trainable variable $w_k \in \boldsymbol{w}$ is given by

$$w_k = w_{k-1} - \alpha_k m_k \tag{37}$$

with

$$\alpha_k = \frac{\alpha}{\sqrt{v_k} + \epsilon_l}$$

$$m_k = \frac{1}{1 - (\beta_1)^k}\left[\beta_1 m_{k-1} + (1 - \beta_1) g_k\right]$$

$$v_k = \frac{1}{1 - (\beta_2)^k}\left[\beta_2 v_{k-1} + (1 - \beta_2)(g_k)^2\right]$$

$$g_k \overset{\text{def}}{=} \left.\frac{\partial L_{D_k}(\boldsymbol{w})}{\partial w}\right|_{\boldsymbol{w}=\boldsymbol{w}_{k-1}} \tag{38}$$

where β_1 and β_2 are decay rates for the first and second-order moment estimates of stochastic gradient whose recommended values are $\beta_1 = 0.9$ and $\beta_2 = 0.999$ [20]; m_k and v_k are the first and second order moment estimates of stochastic gradient at k^{th} optimization step in each epoch and are initialized by $m_0 = v_0 = 0$; $\epsilon_l = 10^{-8}$ is a small constant for numerical stability; α_k is the learning rate at the current step and α is a hyperparameter used for controlling the learning rate with a recommended value of 0.001 [20].

4. DNN architecture selection and training aspects

This section gives details of the DNN surrogate model together with important training aspects. The implementation and training of DNNs are carried out using Tensorflow [68], while all the FE^2 analyses and homogenization analyses for generating datasets are performed in a Matlab-based in-house finite element library CPSSL-FEA developed by the authors. In all the examples, the Adam optimizer [20] is used with the following *hyperparameters*: decay rates β_1 = 0.9 and β_2 = 0.999, learning rate (α) = 0.001, mini-batch size = 1024, and total number of epochs = 3000. For training parameters initialization, all the biases are initialized to zeros while all weights in convolutional (Conv) and fully-connected (FC) network layers are initialized using the Xavier normal initialization method [15]. With the Xavier method, the variances of DNN output are close to the variance of its input, and this helps to prevent the gradient vanishing or exploding issues during the network training [15].

To evaluate the overall performance of different trained DNN models, mean squared error is used for computing the training/testing error over the complete training/testing dataset. Moreover, for generating the overall error statistics, for each testing sample, if the target stress is $\boldsymbol{y}$ and the network prediction is $\boldsymbol{y}$, the prediction performance of the network is evaluated by using the following relative error metric

$$\varepsilon_r = \frac{\|\boldsymbol{y} - \boldsymbol{y}\|_2}{\|\boldsymbol{y}\|_2} \times 100(\%) \tag{39}$$

4.1. Training/Testing datasets

The input of the DNN model is $\bar{\boldsymbol{C}}$ which, for 2D plane strain, can be parameterized by

$$\bar{\boldsymbol{C}} = \bar{\boldsymbol{U}}^2 \text{ with } \bar{\boldsymbol{U}} = \boldsymbol{Q}\bar{\boldsymbol{\Lambda}}\boldsymbol{Q}^T,\ \boldsymbol{Q}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix},\ \bar{\boldsymbol{\Lambda}} = \begin{bmatrix} \bar{\lambda}_1 & \\ & \bar{\lambda}_2 \end{bmatrix} \tag{40}$$

where $\bar{\lambda}_1$ and $\bar{\lambda}_2$ are macroscopic principal stretch ratios, θ is the angle of the principal axes w.r.t. the standard Euclidean bases $\{\boldsymbol{e}_1, \boldsymbol{e}_2\}$. Due to symmetry, the training parameters can be chosen as $\bar{\lambda}_1, \bar{\lambda}_2 \in [\bar{\lambda}_{\min}, \bar{\lambda}_{\max}]$ and $\theta \in [0, \pi/2)$ where $\bar{\lambda}_{\min}$ and $\bar{\lambda}_{\max}$ are the lower and upper bounds of the macroscopic stretch ratios that determine the surrogate model application range. In this study, $\bar{\lambda}_{\min}$ = 0.5 and $\bar{\lambda}_{\max}$ = 2 are chosen for the training of the surrogate models and the surrogate models are only applicable within this dataspace.

4.1.1. Training data generation

For the efficiency of training data generation, the Latin Hypercube Sampling (LHS) [69] is carried out in parameter space $\{\theta, \alpha\} \in [0, \pi/2) \times [0, 2\pi)$ where θ represents the angle of principal axes in Eq. (40) and α is the sweeping angle in the principal stretch ratios space, see Fig. 7 where a loading path is described by a line starting from the undeformed state $(\bar{\lambda}_1 = \bar{\lambda}_2 = 1)$ with an inclined angle α and ending at the boundary of the macro stretch ratios space $[0.5,\ 2] \times [0.5,\ 2]$ at point *A*. The loading path is then divided uniformly into a prescribed number of N_l load steps, see the red dots in Fig. 7. As a result, each sampled value of α represents a sampled loading path. The red lines in Fig. 8 illustrate a uniform sampling of α with 50 points in $[0, 2\pi)$ in the macro stretch ratios space. In Fig. 8, each loading path (red line) is then divided uniformly into a number of loading steps (N_l), where N_l is proportional to the length of the total path (O-A). For example, in Fig. 8, N_l is defined such that the shortest loading path (from point (1, 1) to point (1, 0.5) or point (0.5, 1)) has 10 steps and the N_l is proportionally increased for other loading paths. After generating the samples for $\bar{\boldsymbol{C}}$, the corresponding $\bar{\boldsymbol{S}}$ and $\bar{\mathbb{C}}$ are obtained by a homogenization analysis to generate the full training dataset $D_{\text{train}} = \{\bar{\boldsymbol{C}}^{(r)}, \bar{\boldsymbol{S}}^{(r)}, \bar{\mathbb{C}}^{(r)}\}_{r=1}^{m}$. It is noted that in this dataset generation, only one homogenization analysis is carried out for each loading path rather than for each sample. For the validation of a surrogate model during the training process, the dataset is spit in a ratio of 9:1 to two sets – training set and validation set. In this study, the appropriate size of the training dataset is also investigated, see Section 4.3, where for different dataset sizes the corresponding sampling densities in each domain are given in Table 1.

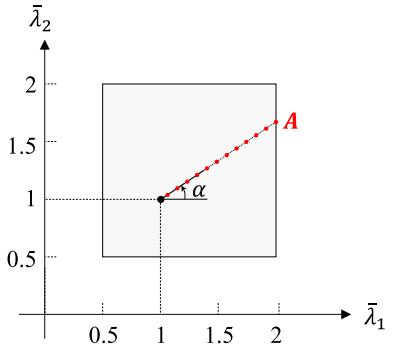


Fig. 7. Parameterization of macroscopic deformation gradient load path.

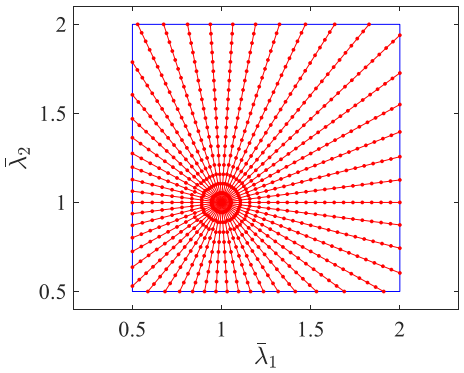


Fig. 8. Illustration of uniformly generated samples in the $(\bar{\lambda}_1, \bar{\lambda}_2)$ space. Blue box denotes the boundary of the data space; red lines denote generated load paths on which the samples lie; red points denote generated samples. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4.1.2. Testing data generation

To evaluate the performance of the trained model in the training dataspace range, i.e., $\bar{\lambda}_1, \bar{\lambda}_2, \theta \in [0.5,\ 2] \times [0.5,\ 2] \times [0,\ \pi/2)$, the input $\bar{\boldsymbol{C}}$ in the testing-dataset is generated by a $n_T = 100 \times 100 \times 100$ uniform grid in the $(\bar{\lambda}_1, \bar{\lambda}_2, \theta)$ parameter space. The data $\{(\bar{\boldsymbol{C}}^{(r)}, \bar{\boldsymbol{S}}^{(r)}, \bar{\mathbb{C}}^{(r)})\}_{r=1}^{n_T}$ in the testing-dataset is then generated by running n_T homogenization analysis, one at each grid point.

4.1.3. Data preprocessing

For the better numerical performance of neural network models, both the input and output of training data are normalized using their statistics. The normalization is carried for each component of the input vector (e.g., $\bar{C}_{ij}$) and output vector (e.g., $\bar{S}_{ij}$) independently. For example, for any component χ (χ can represent $\bar{C}_{ij}$, $\bar{S}_{ij}$ etc.), the normalization is given by

$$\widehat{\chi}^{(r)} = \frac{\chi^{(r)} - \mu_\chi}{\sqrt{V_\chi}}, \qquad r = 1, \cdots, m \tag{41}$$

with

$$\mu_\chi = \mathbb{E}[\boldsymbol{\chi}] \approx \frac{1}{m}\sum_{r=1}^{m}\chi^{(r)} \text{ and } V_{\boldsymbol{\chi}} = \text{Var}[\boldsymbol{\chi}] \approx \frac{1}{m}\sum_{r=1}^{m}\left(\chi^{(r)} - \mu_\chi\right)^2 \tag{42}$$

where $\boldsymbol{\chi}$ is the vector of training data for component χ, i.e. $\boldsymbol{\chi} = \begin{bmatrix} \chi^{(1)} & \cdots & \chi^{(m)} \end{bmatrix}^T$ where m is the total number of training samples. The normalized training dataset, $\widehat{D}_{\text{train}}$, contains all the normalized components $\widehat{\chi}$. The testing dataset is also normalized by the statistics from the training dataset (μ_χ and $V_{\boldsymbol{\chi}}$ in Eq. (42)) using Eq. (41).

4.1.4. Scaling strategies for network derivatives

In this study, two scaling strategies for the network derivatives used in the Sobolev training are considered – one uses full training dataset statistics and the other one uses mini-batch dataset statistics. Since the derivatives are only used for training and not for the model evaluation, the preprocessing of the derivative data can be made during training without storing the statistics.

(a) 1st scaling strategy

For the 1st scaling strategy, the preprocessing of the derivatives follows Eqs. (41) and (42) in Section 4.1.3 with χ replaced by half of

Table 1
Sampling densities in each domain and the number of samples in the shortest loading path for different training datasets.

Size of dataset	0.5×10^6	10^6	1.5×10^6	2×10^6	2.5×10^6	3×10^6
Samples in (α, θ) by LHS	1.8×10^4	3.6×10^4	5.4×10^4	7.2×10^4	9×10^4	1.08×10^5
Steps in shortest loading path	20	20	20	20	20	20

$\bar{\mathbb{C}}$ (i.e., the target mapping derivative J in Eq. (30)). Due to the preprocessing of the input and output data, the calculation of the normalized approximate derivative given by DNN is obtained through chain rule by

$$\left(\widehat{\frac{\partial \bar{S}_{ij}}{\partial \bar{C}_{kl}}}\right) = \frac{\sqrt{\frac{\mathrm{Var}[\bar{S}_{ij}]}{\mathrm{Var}[\bar{C}_{kl}]}}\frac{\partial \widehat{\bar{S}}_{ij}}{\partial \widehat{\bar{C}}_{kl}} - \mathbb{E}\left[\frac{1}{2}\bar{\mathbb{C}}_{ijkl}\right]}{\sqrt{\mathrm{Var}\left[\frac{1}{2}\bar{\mathbb{C}}_{ijkl}\right]}} \tag{43}$$

where the statistics such as expectation $\mathbb{E}[\ ]$ and variance $\mathrm{Var}[\ ]$ of those quantities $\bar{S}_{ij}$, $\bar{C}_{kl}$ and $\bar{\mathbb{C}}_{ijkl}$ are computed over the full training dataset (see Eq. (42)). It is noted that the term $\partial \widehat{\bar{S}}_{ij}/\partial \widehat{\bar{C}}_{kl}$ represents the derivative of the DNN model, while the calculated term $\left(\widehat{\frac{\partial \bar{S}_{ij}}{\partial \bar{C}_{kl}}}\right)$ is the one used in the loss function in Eq. (30). In this case, the statistics of $\bar{\mathbb{C}}_{ijkl}$ are pre-calculated before training process.

(b) 2nd scaling strategy

In the 2nd scaling strategy, the preprocessing of the derivatives data is done for each mini-batch separately during the training process, i.e., the statistics of the derivatives are computed over s_b samples inside each batch by

$$\begin{aligned}\mathbb{E}_{s_b}\left[\frac{1}{2}\bar{\mathbb{C}}_{ijkl}\right] &\approx \frac{1}{s_b}\sum_{r=1}^{s_b}\frac{1}{2}\bar{\mathbb{C}}_{ijkl} \ \text{ and } \ \mathrm{Var}_{s_b}\left[\frac{1}{2}\bar{\mathbb{C}}_{ijkl}\right] \\ &\approx \frac{1}{s_b}\sum_{r=1}^{s_b}\left(\frac{1}{2}\bar{\mathbb{C}}^{(r)}_{ijkl} - \mathbb{E}_{s_b}\left[\frac{1}{2}\bar{\mathbb{C}}_{ijkl}\right]\right)^2\end{aligned} \tag{44}$$

where $\bar{\mathbb{C}}^{(r)}_{ijkl}$ are derivative data from mini-batch training dataset D_k, and hence the calculation of the normalized approximate derivative given by DNN is obtained by

$$\left(\widehat{\frac{\partial \bar{S}_{ij}}{\partial \bar{C}_{kl}}}\right) = \frac{\sqrt{\frac{\mathrm{Var}[\bar{S}_{ij}]}{\mathrm{Var}[\bar{C}_{kl}]}}\frac{\partial \widehat{\bar{S}}_{ij}}{\partial \widehat{\bar{C}}_{kl}} - \mathbb{E}_{s_b}\left[\frac{1}{2}\bar{\mathbb{C}}_{ijkl}\right]}{\sqrt{\mathrm{Var}_{s_b}\left[\frac{1}{2}\bar{\mathbb{C}}_{ijkl}\right]}} \tag{45}$$

4.2. DNN architecture candidates

Fig. 9 shows the basic neural network architecture that is used to describe the mean function $\mathcal{F}: x \to y$ in Eq. (19). Between input $\boldsymbol{x}$ (i.e., $\bar{\boldsymbol{C}}$) and output $\boldsymbol{y}$ (i.e., $\bar{\boldsymbol{S}}$), there are K blocks, each of which is a combination of one or more different types of neural network operation layers. Specifically, four types of neural network blocks are considered in this study (see Fig. 10). In Fig. 10, *block type 1* represents fully-connected (FC) layer; *block type 2* represents FC layer with ResNet connection [12], where it is required that the first neural network block in Fig. 9 has to be an FC layer since ResNet can only be applied to layers of the same input and output dimensions; *block type 3* contains FC layer and convolutional (Conv) layer [6]; while *block type 4* and *block type 5* include FC, Conv, and ResNet layers. The difference between block type 4 and block type 5 is the relative positions of the FC and Conv layers. These layers are chosen since FC and Conv layers provide approximation capacity by introducing trainable parameters, and ResNet connection provides shortcuts in backpropagation process, which can prevent gradient vanishing or exploding issues in training stage. The details on the mathematical operations in FC, Conv and ResNet connections are provided in the following subsections.

4.2.1. Fully connected layer

Suppose the l th fully connected layer $\boldsymbol{x}^{[l]}$ contains $n_H^{[l]}$ neurons with the input from the last layer $\boldsymbol{x}^{[l-1]}$ of size $n_H^{[l-1]} \times 1$, the mathematical operation in this layer reads

$$\boldsymbol{x}^{[l]} = \begin{cases} \boldsymbol{h}_1^{[l]} & \text{if layer } l \text{ is the last layer} \\ \Phi\left(\boldsymbol{h}_1^{[l]}\right) & \text{otherwise} \end{cases} \tag{46}$$

$$\boldsymbol{h}_1^{[l]} = \boldsymbol{W}^{[l]}\boldsymbol{x}^{[l-1]} + \boldsymbol{b}^{[l]} \tag{47}$$

where $\boldsymbol{W}^{[l]}$ is the weight matrix of size $n_H^{[l]} \times n_H^{[l-1]}$ and $\boldsymbol{b}^{[l]}$ is the bias vector of size $n_H^{[l]} \times 1$. As a result, vector $\boldsymbol{h}_1^{[l]}$ is of size $n_H^{[l]} \times 1$. For intermediate layers, nonlinear activation functions denoted as $\Phi(\cdot)$, are used to create nonlinear mappings using element-wise scalar operation. Here, ReLU activation function is adopted, i.e. $\Phi(\cdot) = \Phi_{\mathrm{ReLU}}(\cdot)$, which is defined as

$$\Phi_{\mathrm{ReLU}}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{48}$$

4.2.2. Convolutional (Conv) layer

Suppose l th layer is a Conv layer, then operations including padding, convolution, and activation are applied sequentially to the input $\boldsymbol{x}^{[l-1]}$ of dimension $n_H^{[l-1]} \times n_C^{[l-1]}$. Here, an equal padding operation [70] is used, which means that zero blocks are padded on the top and bottom sides of the volume $\boldsymbol{x}^{[l-1]}$ such that the height of the volume after the following convolution operation remains the same as $\boldsymbol{x}^{[l-1]}$, i.e. $n_H^{[l-1]}$. The output of the padding operation is denoted as $\boldsymbol{h}_1^{[l]} = [0; \boldsymbol{x}^{[l-1]}; 0]$ where the size of each zero block is denoted as $p^{[l]} \times n_C^{[l-1]}$. As a result, the size of $\boldsymbol{h}_1^{[l]}$ is $\left(2p^{[l]} + n_H^{[l-1]}\right) \times n_C^{[l-1]}$. The determination of $p^{[l]}$ is discussed later.

After padding, the convolution operation is applied to the volume $\boldsymbol{h}_1^{[l]}$ using $n_C^{[l]}$ filters. The size of each filter is $f^{[l]} \times n_C^{[l-1]}$ and the filter stride used is denoted by $s^{[l]}$. In each filter, there is a weight matrix $\boldsymbol{W}_i^{[l]}$ and bias $b_i^{[l]}$ ($i = 1, 2, \cdots, n_C^{[l]}$). With the i th filter, the convolution operation outputs a vector

$$\boldsymbol{v}_i = \boldsymbol{h}_1^{[l]} * \boldsymbol{W}_i^{[l]} = \begin{bmatrix} \boldsymbol{h}_1^{[l]}\left[1 : f^{[l]}\right] \star \boldsymbol{W}_i^{[l]} + b_i^{[l]} \\ \boldsymbol{h}_1^{[l]}\left[s^{[l]}+1 : s^{[l]}+f^{[l]}\right] \star \boldsymbol{W}_i^{[l]} + b_i^{[l]} \\ \boldsymbol{h}_1^{[l]}\left[2s^{[l]}+1 : 2s^{[l]}+f^{[l]}\right] \star \boldsymbol{W}_i^{[l]} + b_i^{[l]} \\ \vdots \\ \boldsymbol{h}_1^{[l]}\left[2p^{[l]}+n_H^{[l-1]}-f^{[l]}+1 : 2p^{[l]}+n_H^{[l-1]}\right] \star \boldsymbol{W}_i^{[l]} + b_i^{[l]} \end{bmatrix} \tag{49}$$

with dimension

$$\left(\frac{2p^{[l]} + n_H^{[l-1]} - f^{[l]}}{s^{[l]}} + 1\right) \times 1 \tag{50}$$

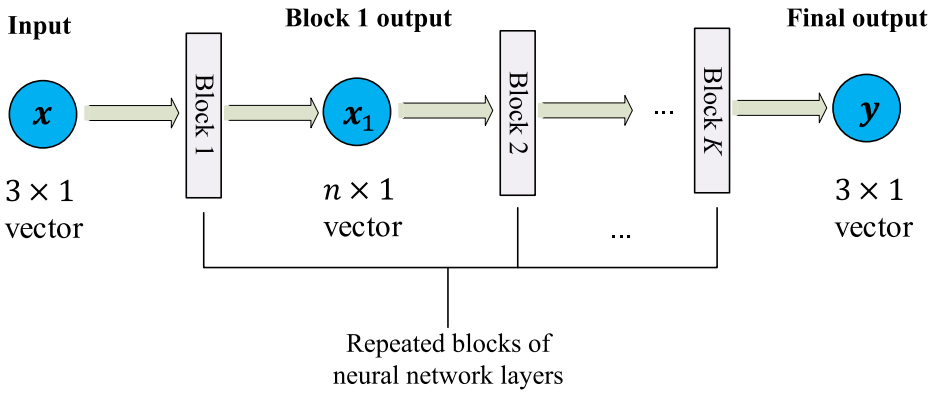


Fig. 9. Sketch of neural network architecture.

and $\boldsymbol{h}_1^{[l]}[p:q]$ means that the volume $\boldsymbol{h}_1^{[l]}$ is sliced along the height from p to q. The operation $\star$ applies to two volumes $\boldsymbol{a}$ and $\boldsymbol{b}$ of the same size, denoted as $n_1 \times n_2$, and is given by

$$\boldsymbol{a}\star\boldsymbol{b} \stackrel{\text{def}}{=} \sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\boldsymbol{a}[i,j]\boldsymbol{b}[i,j] \tag{51}$$

Finally, the vectors $\boldsymbol{v}_i$ $(i = 1, 2, \cdots, n_C^{[l]})$ are stacked together to output a new volume

$$\boldsymbol{h}_2^{[l]} = \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots & \boldsymbol{v}_{n_C^{[l]}} \end{bmatrix} \tag{52}$$

which is of size

$$\left(\frac{2p^{[l]} + n_H^{[l-1]} - f^{[l]}}{s^{[l]}} + 1\right) \times n_C^{[l]} \tag{53}$$

By the definition of equal padding, it is required that the height of $\boldsymbol{h}_2^{[l]}$ is equal to the height of $\boldsymbol{x}^{[l-1]}$. Thus, the padding size can be determined as

$$p^{[l]} = \frac{1}{2}\left[\left(n_H^{[l-1]} - 1\right)s^{[l]} + f^{[l]} - n_H^{[l-1]}\right] \tag{54}$$

As a result, the size of the volume $\boldsymbol{h}_2^{[l]}$ becomes $n_H^{[l-1]} \times n_C^{[l]}$.

To add nonlinearities in the layer, elementwise ReLU activation function is applied to the volume $\boldsymbol{h}_2^{[l]}$ and results in the output of the Conv layer as

$$\boldsymbol{x}^{[l]} = \Phi_{\text{ReLU}}\left(\boldsymbol{h}_2^{[l]}\right) \tag{55}$$

4.2.3. ResNet connections

When the ResNet connection is applied between l th layer and $(l+s)^{\text{th}}$ layer where s is a positive integer, the input of $(l+s+1)^{\text{th}}$ layer is obtained as

$$\boldsymbol{x}_{in}^{[l+s+1]} = \boldsymbol{x}_{out}^{[l+s]} + \boldsymbol{x}_{in}^{[l]} \tag{56}$$

where $\boldsymbol{x}_{in}^{[l]}$ is the input of l th layer and $\boldsymbol{x}_{out}^{[l+s]}$ is the output of $(l+s)^{\text{th}}$ layer.

4.3. Training data size – Regular training

The training data size depends on the complexity of the underlying mapping that must be learned and may vary for the considered problems. To determine the appropriate size of the training dataset, multiple training datasets of different sizes, i.e., m$\in\{0.5 \times 10^6, 1 \times 10^6, 1.5 \times 10^6, 2 \times 10^6, 2.5 \times 10^6, 3 \times 10^6\}$ (Table 1), are examined for training a fully connected feedforward DNN surrogate with three hidden layers and 80 neurons in each hidden layer. This DNN surrogate is trained for the unit cell-1 shown in Fig. 4. The mean squared training and testing errors are shown in Fig. 11, where it can be observed that the training and testing error decreases with the increase in the size of the datasets. However, improvements in the accuracy are not significant when the dataset size increases from 2.5 million to 3 million. From this pilot study, the training dataset size is fixed at 2.5 million for all the cases considered in this paper.

4.4. DNN architecture exploration – Regular training

Although a feedforward DNN surrogate with three layers and 80 neurons in each layer provides good accuracy when an appropriate dataset is used, a further investigation on the DNN architecture is carried out to find surrogates with higher computational efficiency and accuracy. To this end, first, some hyperparameters such as the number of hidden layers and the number of neurons in each layer are investigated. This task is carried out using DNN with *block type 1*, i.e., feedforward neural network. The number of hidden layers n_l and the number of neurons in each layer n_p are chosen from the sets {1, 2, 3, 4, 5} and {20, 40, 60, 80, 100}, respectively. The testing errors for the 25 feedforward neural networks are shown in Fig. 12. Considering a balance between computational efficiency and accuracy, the combination of n_l = 3 and n_p = 80 is chosen for FC layers.

As shown in Fig. 10, more expressive capacity in the DNN architecture can be achieved by adding Conv layers and/or ResNet connections. To investigate if the added capacity can be useful in improving the performance of DNN, architectures considering *block type 2, 3*, 4 and *5* are trained with the same training dataset and optimization setup mentioned in Section 4. For the four different types of architectures, three (K = 3) repeated blocks and FC layer with 80 neurons are considered. For *block types 3–5*, the filter size and stride in the Conv layers are fixed to be 2×1 and 1, respectively, and the number of filters is chosen as 32, which is the optimal number from a grid search from the set {8, 16, 32, 64, 128}.

The comparison of different architectures in terms of the relative testing errors are given in Table 3 and Fig. 13. As can be seen, the network architectures with *block type 4* and *5* have similar testing performance and give the larger number of testing samples with relative errors smaller than different thresholds and the smallest upper bound of all testing relative errors as compared to the other three architectures. The results show that adding ResNet connections or Conv layers to feedforward neural networks helps to improve the neural network capacity and results in lower error statistics. Besides, these special architecture features do not introduce many additional model parameters as compared to when an FC layer is added. As a result, the neural network architectures con-

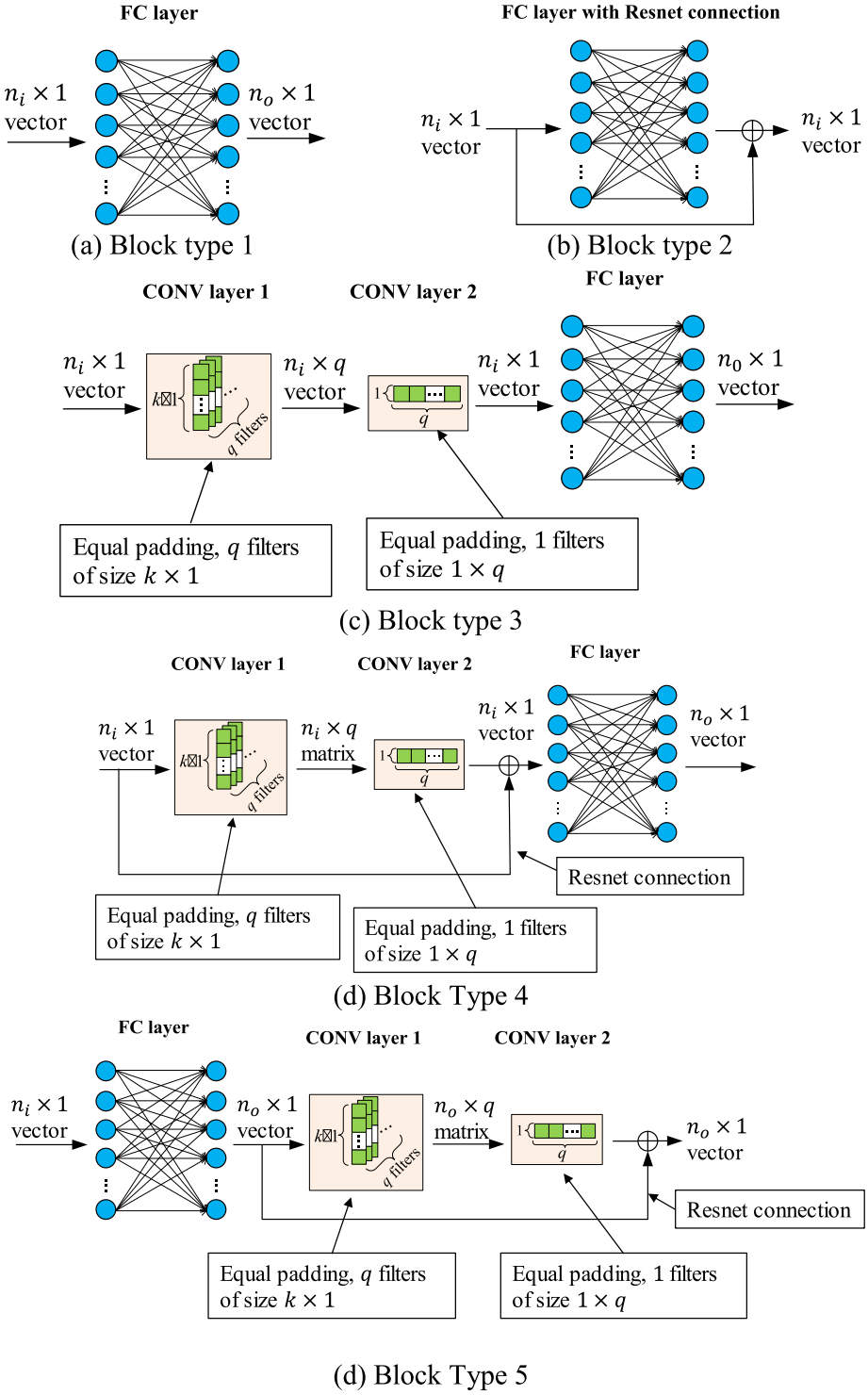


Fig. 10. Five types of neural network blocks.

sidering *block type 4* and *5* are chosen as the final candidate models to be trained for replacing the homogenization analysis. The total number of trainable parameters is provided in Table 2.

4.5. Sobolev training vs. regular training

As discussed in Section 3.4, when the derivatives of the target output w.r.t. the inputs are accessible, the use of Sobolev training in the loss function can be beneficial. For Sobolev training, the final neural network architectures with *block type 4* and *5* in Section 4.4 are trained using the loss function in Eq. (30) with the weighting factor $\lambda_p = 8 \times 10^{-4}$. The weighting factor is determined through a grid search from the set $\{10^{-1}, 10^{-2}, 10^{-3}, 8 \times 10^{-4}, 5 \times 10^{-4}, 10^{-4}, 10^{-5}\}$. The derivatives, i.e., $\boldsymbol{J} = \bar{\mathbb{C}}$ in Eq. (30), are obtained from homogenization analysis (see Appendix A). It is observed that the training error drops from 2.76×10^{-6} (regular) to 1.19×10^{-6} (Sobolev with 1st scaling strategy) or 1.06×10^{-6} (Sobolev with 2nd scaling strategy) and the testing error drops from

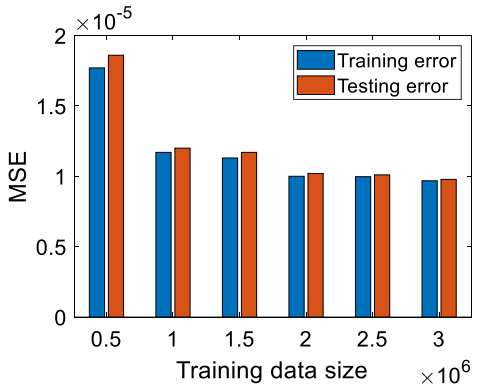


Fig. 11. Training and testing mean squared errors with different training data sizes.

2.78×10^{-6} (regular) to 1.20×10^{-6} (Sobolev with 1st scaling strategy) or 1.06×10^{-6} (Sobolev with 2nd scaling strategy). A detailed comparison of the relative testing error statistics is given in Table 4 and Table 5, where it shows a higher percentage of relative errors below 2%, 1%, and 0.5% with Sobolev training as compared to regular training. Besides, it can be found that when Sobolev training is used, the architecture considering *block type 5* and 2nd scaling strategy for network derivatives has more testing samples with small relative errors as compared to the other three cases where either architecture with *block type 4* or 1st scaling strategy is considered. In Figs. 14 and 15, box plots are used to show the overall distribution of the relative testing error statistics with Sobolev and regular training. On each box, the upper and lower bounds represent the worst and best relative error, respectively, and the bottom and top edges of the box indicate the 25th and 75th percentiles, respectively, and the central mark denotes the median. From Figs. 14 and 15, it can be observed that Sobolev training leads to smaller values of upper bound, median, 25th· and 75th percentiles of the relative testing error statistics than regular training, which again confirms the better testing performance of Sobolev training. Moreover, when Sobolev training is used, the network architecture with *block type 5* and 2nd scaling strategy gives the best performance in terms of testing error statistics. Hence, the architecture with *block type 5* using Sobolev training and 2nd scaling strategy for network derivative will be adopted in all the following examples.

4.6. FE^2 tests on one macro-element

In this section, DNN surrogate models with *block type 5* in Fig. 10 are constructed and trained with Sobolev loss function and 2nd scaling strategy for two different unit cells. Although the determination of the training data size, as well as the hyperparameters of the DNN, are only studied on unit cell-1 (Section 3.1.1), the same settings are adopted for the DNN surrogate for unit cell-2 (Section 3.1.2), as well. Results in Section 4.6.2 serve as the justification of this choice, where the satisfactory performance of the constructed DNN surrogate is observed.

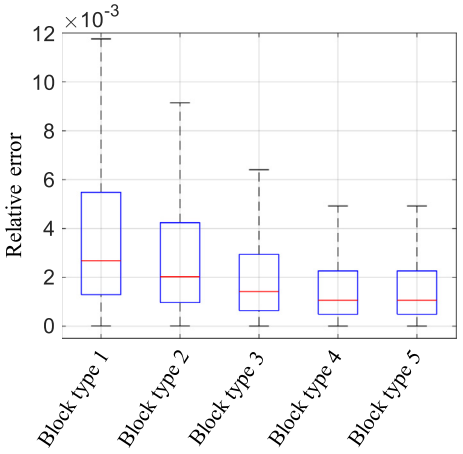


Fig. 13. Box plot of relative errors of testing data points for different neural networks.

Table 2
Number of trainable parameters for different network architectures.

Network type	Number of trainable parameters
1	13,523
2	13,523
3	14,006
4	14,006
5	14,006

Table 3
The percentage of testing samples with relative errors smaller than different thresholds for different neural network models.

Block type	5% threshold	2% threshold	1% threshold	0.5% threshold
1	99.98%	97.79%	89.31%	72.14%
2	99.98%	98.99%	93.02%	79.53%
3	99.99%	99.85%	97.23%	87.90%
4	99.99%	99.93%	98.88%	92.13%
5	99.99%	99.92%	98.69%	92.11%

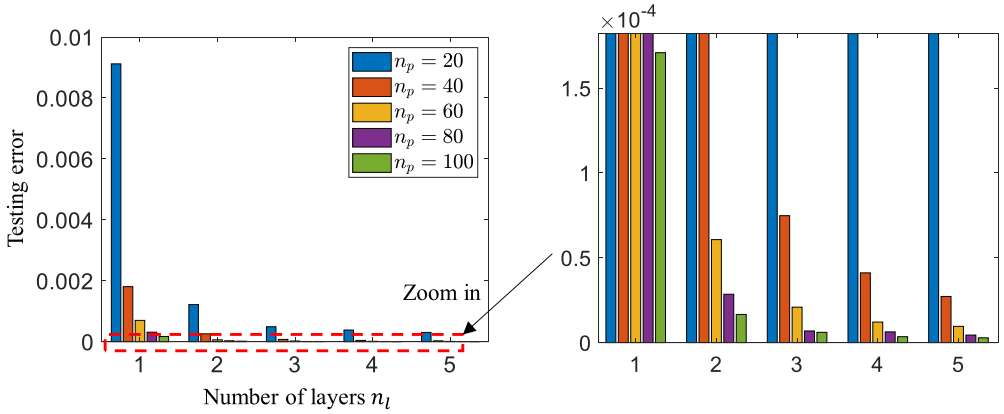


Fig. 12. Testing errors of feedforward neural networks with a different number of layers and neurons.

Table 4
The percentage of testing samples with relative errors smaller than different thresholds for regular and Sobolev training of DNN with *block type 4* on unit cell-1.

Training	5% threshold	2% threshold	1% threshold	0.5% threshold
Regular	99.99%	99.93%	98.88%	92.13%
Sobolev + 1st scaling strategy	99.99%	99.97%	99.37%	94.84%
Sobolev + 2nd scaling strategy	99.99%	99.97%	99.38%	94.88%

Table 5
The percentage of testing samples with relative errors smaller than different thresholds for regular and Sobolev training of DNN with *block type 5* on unit cell-1.

Training	5% threshold	2% threshold	1% threshold	0.5% threshold
Regular	99.99%	99.92%	98.69%	92.11%
Sobolev + 1st scaling strategy	99.99%	99.97%	99.37%	94.86%
Sobolev + 2nd scaling strategy	99.99%	99.97%	99.55%	95.74%

4.6.1. Unit cell-1

The DNN surrogate of the unit cell-1 in Section 3.1.1 is trained with the 2.5 million training dataset. The error statistics on the testing dataset are shown in Fig. 15 and Table 5.

4.6.1.1. Uniaxial tension and compression. To demonstrate how the trained DNN surrogate performs as compared to the direct homogenization result, uniaxial tension and compression test is carried out on one element with 4 integration points. The test settings are shown in Fig. 16a, where u = -30 for compression and u = 100 for tension. The load–displacement curves generated using the trained DNN model and direct homogenization result are compared in Fig. 17a, where a close match can be observed. The macroscopic stretch ratios at all integration points during the loading process are plotted together with the data range of the training space in Fig. 17b, where all the data points are bounded by the training data range, as intended.

4.6.1.2. Simple shear. The second single element test is a simple shear test, shown in Fig. 16b, where u = 150 is used. The load–displacement curves from the trained DNN model and homogenization result are shown in Fig. 18a, where a close match is again observed. The macroscopic stretch ratios at all integration points during the loading process are plotted in Fig. 18b, where all the points are again bounded by the training data range.

4.6.2. Unit cell-2

The same architecture of the DNN, as used for unit cell-1 and given in Section 4.4, is again chosen for unit cell-2 and is trained using regular and Sobolev training with the same weighting factor, i.e., $\lambda_p = 8 \times 10^{-4}$, and 2nd scaling strategy for network derivatives using 2.5 million training samples. Comparisons of the regular and Sovolev training in terms of statistics of the testing errors are shown in Fig. 19 and Table 6. Compared to regular training, Sobolev training leads to a larger number of testing samples with relative errors smaller than different thresholds, and smaller upper bound and median of all testing relative errors. Thus, Sobolev training gives a better testing performance than regular training in this test case as well.

4.6.2.1. Uniaxial tension and compression. The uniaxial tension and compression test in Fig. 16a are carried out again for the unit cell in Fig. 5 with u = -30 for compression and u = 100 for tension. The comparison of the load–displacement curves from the trained DNN model and homogenization result is given in Fig. 20a, where a close match can be observed. The macroscopic stretch ratios at all integration points during the loading process are plotted in Fig. 20b which shows that all the points are bounded by the training data range.

4.6.2.2. Simple shear. The simple shear test shown in Fig. 16b with u = 150 is again checked for unit cell-2 in Fig. 5. Fig. 21a shows the load–displacement curves from the trained DNN model and homogenization result, where a close match can be seen. Fig. 21b plots the macroscopic stretch ratios at all integration points during

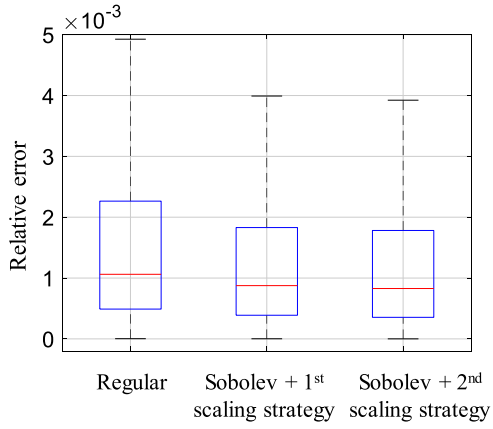


Fig. 14. Box plot of relative testing errors with regular and Sobolev training of DNN with *block type 4* for unit cell-1.

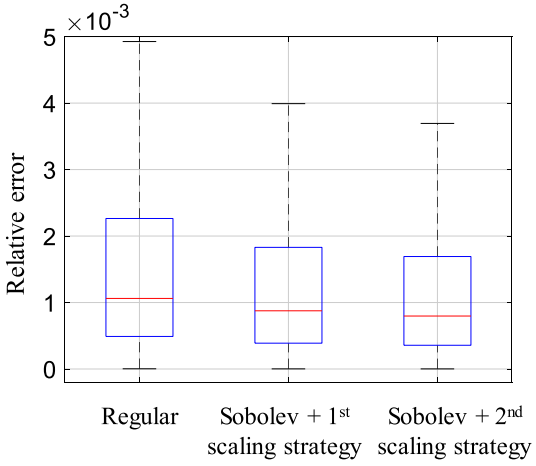
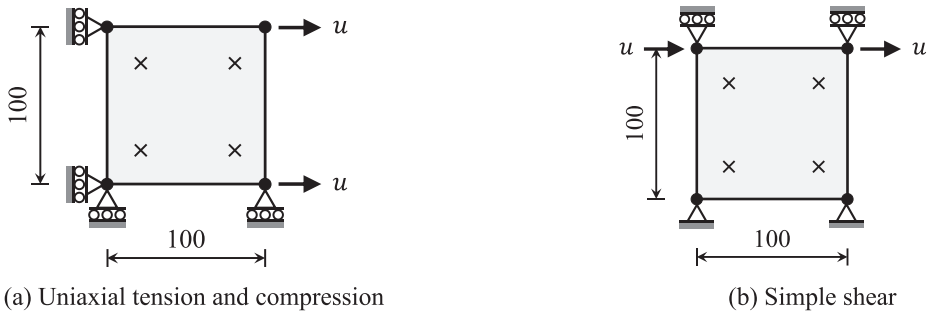
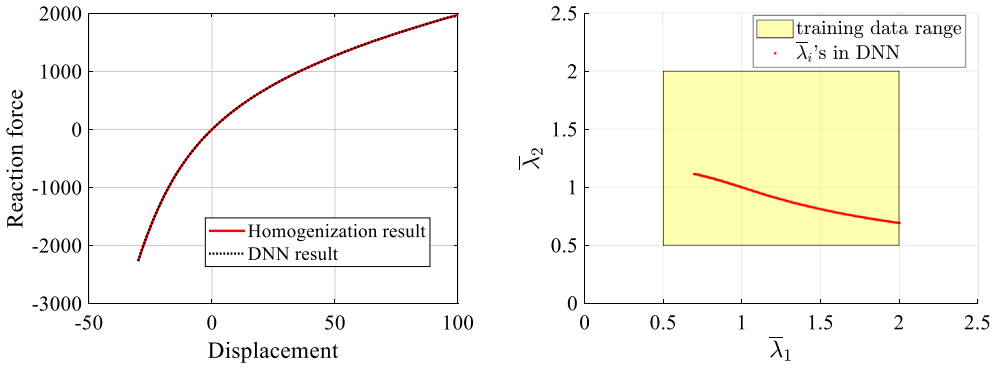


Fig. 15. Box plot of relative testing errors with regular and Sobolev training of DNN with *block type 5* for unit cell-1.



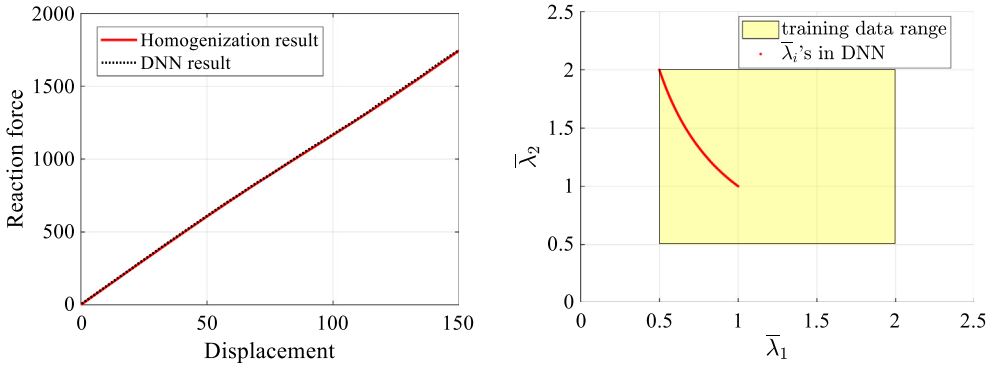
(a) Uniaxial tension and compression (b) Simple shear

Fig. 16. One quadrilateral element test.



(a) Load-displacement curves (b) Stretch ratios during the loading process (DNN)

Fig. 17. Uniaxial tension and compression test results for unit cell-1.



(a) Load-displacement curves (b) Stretch ratios during the loading process (DNN)

Fig. 18. Simple shear test results for unit cell-1.

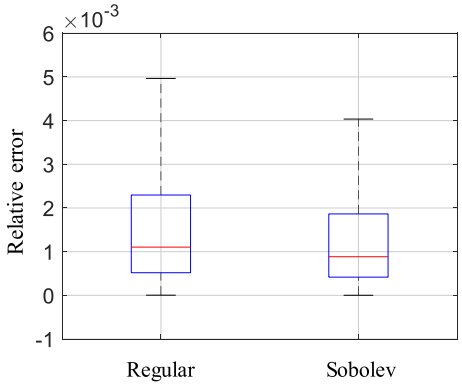


Fig. 19. Box plot of relative testing errors with regular and Sobolev training and 2nd scaling strategy for unit cell-2.

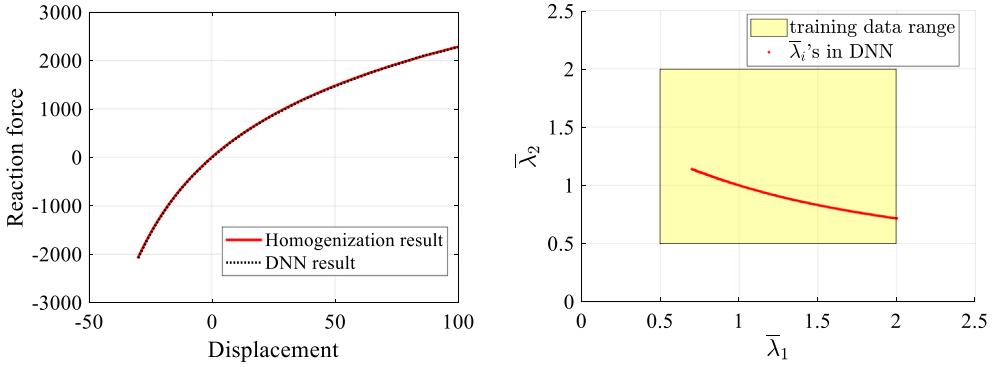
the loading process, where all the points are bounded by the training data range, as desired.

4.7. Computational efficiency

To evaluate the computational efficiency of the DNN surrogate model when compared to direct homogenization analysis, a loading path corresponding to shear deformation is examined at an integration point for both unit cells in Section 4.6. To this end, a single integration point is considered in a macro element that undergoes a deformation gradient $\bar{\boldsymbol{F}} = \bar{\boldsymbol{U}}$, with $\bar{\boldsymbol{U}}$ parameterized by $\bar{\lambda}_1 = 1 + 2\lambda$ and $\bar{\lambda}_2 = 1 - \lambda$ with $\theta = 0°$ or $45°$, see Eq. (40). The CPU time consumptions with DNN surrogate and direct homogenization analyses for different λ and θ are plotted for comparison with unit cell-1 in Fig. 22, and with unit cell-2 in Fig. 23. For both unit cells, the CPU time for DNN evaluation does not depend on the

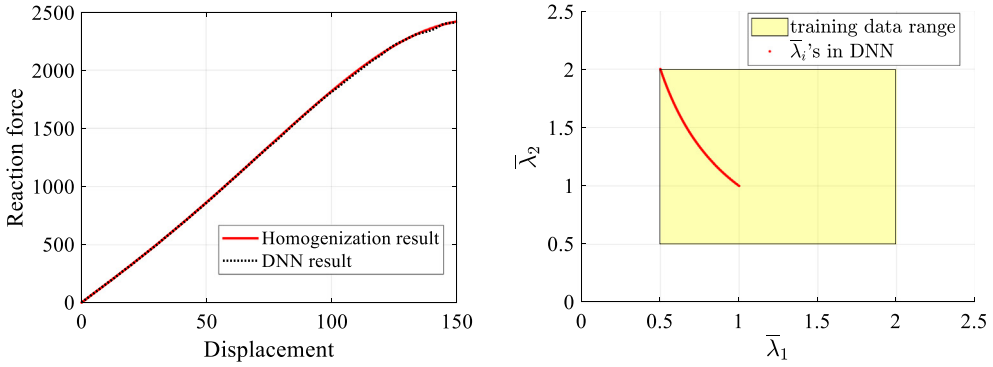
Table 6
The percentage of testing samples with relative errors smaller than different thresholds for regular and Sobolev training of DNN with *block type 5* on unit cell-2.

Training	5% threshold	2% threshold	1% threshold	0.5% threshold
Regular	99.99%	99.91%	98.62%	92.04%
Sobolev + 2nd scaling strategy	99.99%	99.96%	99.28%	94.52%



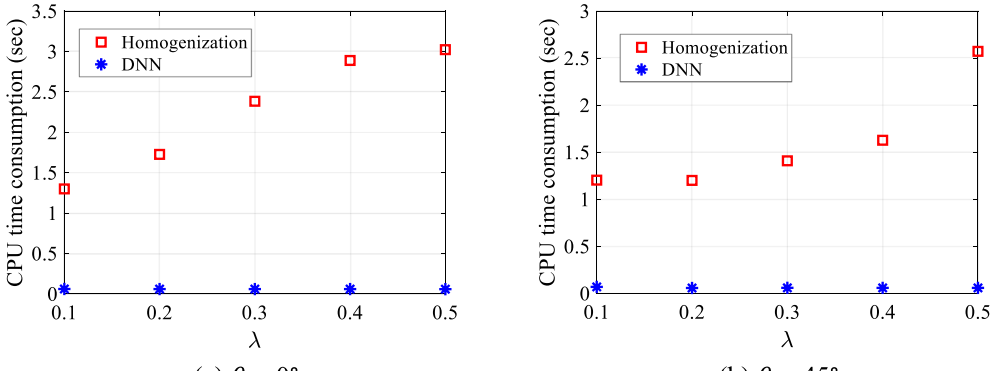
(a) Load-displacement curves (b) Stretch ratios during the loading process (DNN)

Fig. 20. Uniaxial tension and compression test results for unit cell-2.



(a) Load-displacement curves (b) Stretch ratios during the loading process (DNN)

Fig. 21. Simple shear test results for unit cell-2.



(a) $\theta = 0°$ (b) $\theta = 45°$

Fig. 22. CPU time from homogenization analysis and DNN evaluation for unit cell-1.

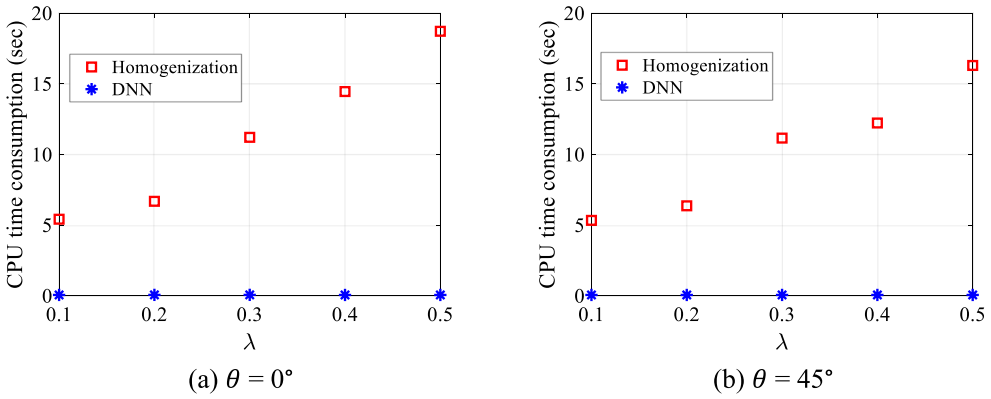


Fig. 23. CPU time from homogenization analysis and DNN evaluation for unit cell-2.

deformation λ and is always around 0.06 sec, while the CPU time for direct homogenization analysis depends on the deformation, since higher deformation usually needs more Newton-Raphson iterations to converge and may also require multiple loading steps. The CPU time difference between DNN and homogenization analysis can be much higher when the unit cell finite element model is big and expensive to evaluate, see unit cell-2 in Fig. 5 where a dense mesh is needed to capture the complex geometry. In Fig. 23, the CPU time for homogenization analysis is about 100 times to 400 times longer than that for DNN evaluation depending on the deformation λ.

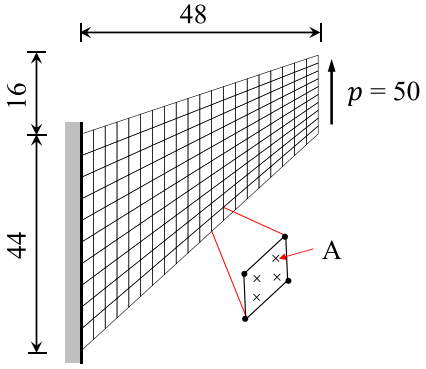


Fig. 24. Geometry and FE mesh of the (macroscale) Cook's membrane problem.

5. FE^2 numerical tests

To show the effectiveness of using the trained surrogate in FE^2 analysis, two macroscale BVPs are considered with the unit cells given in Sections 4.6.1 and 4.6.2.

5.1. Cook's membrane with unit cell-1

5.1.1. No rotation of microstructure

The Cook's membrane macroscale problem sketched in Fig. 24 is considered with the microstructure and unit cell in Fig. 4. The macroscopic displacement-load curves obtained from using DNN and direct homogenization analysis in FE^2 are compared in Fig. 25a, where a close match can be observed. The stretch ratios at all macroscale integration points during the loading process from DNN based analysis in Fig. 25b confirms that the input of DNN stays well within the training data range, as required. Next, the macroscale deformation gradient field and 1st PK stress field from the DNN based and homogenization based FE^2 analyses are compared in Figs. 26 and 27, where good matches can be seen. Fig. 28 shows the histograms of relative errors of the macroscopic deformation gradient and 1st PK stress fields. The upper bounds of the relative errors of the two fields are 0.11% and 3.27%, respectively, for macroscopic deformation gradient and 1st PK stress. The maximum errors correspond to $\bar{\boldsymbol{F}} = [0.6851 \quad 0.7520 \quad -0.6423 \quad 0.8043]^T$ and $\bar{\boldsymbol{P}} = [-0.1239 \quad 0.3163 \quad -0.0696 \quad 0.4007]^T$, respectively. Fig. 29 shows the microscopic deformation gradient and 1st PK stress fields at the macroscale integration point $\boldsymbol{A}$ in Fig. 24 at the last

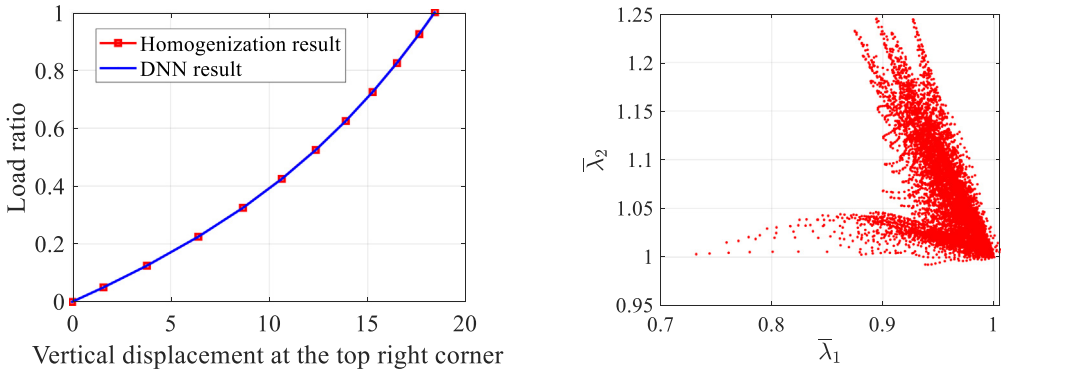


Fig. 25. Results of Cook's membrane with DNN and homogenization analysis on unit cell-1 (no microstructure rotation).

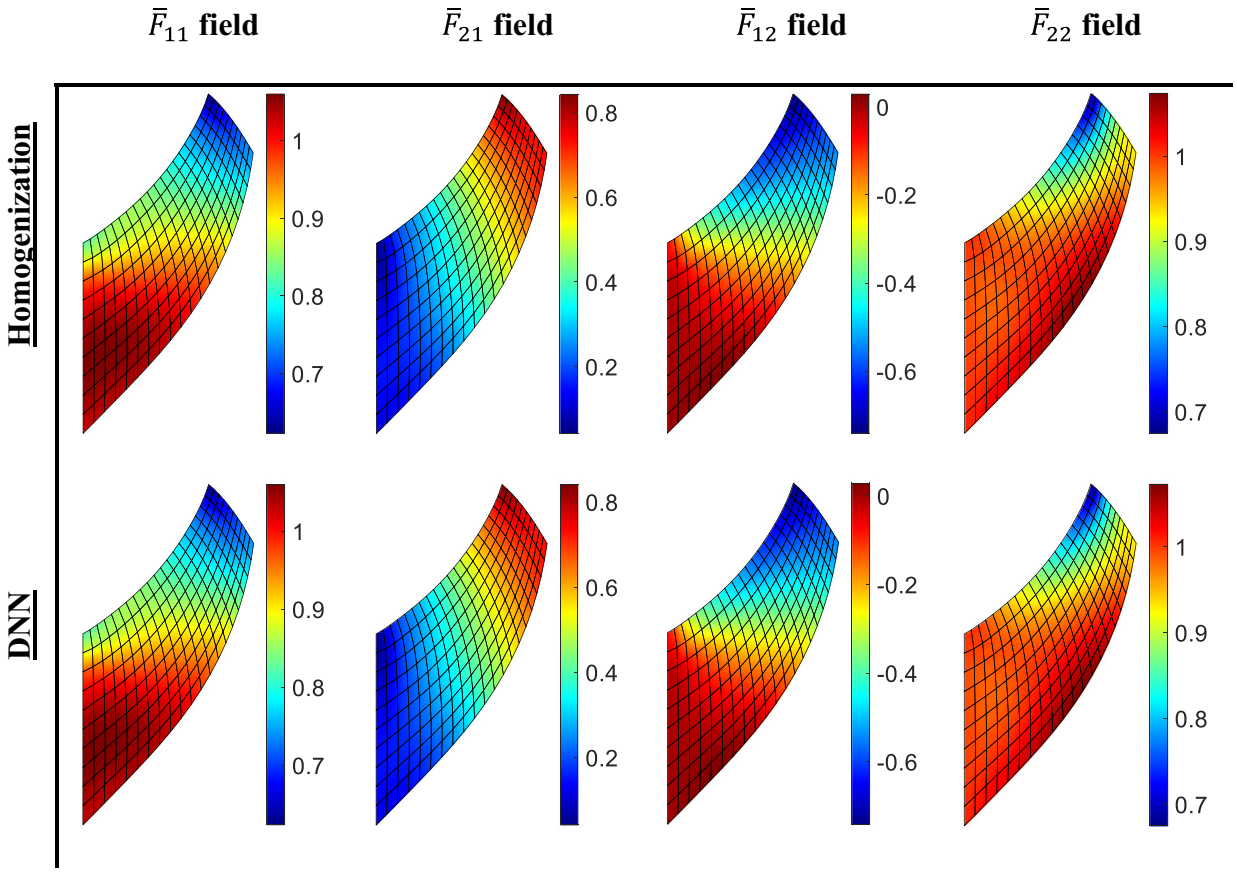


Fig. 26. Macroscopic deformation gradient fields from DNN and homogenization based FE^2 analyses (no microstructure rotation).

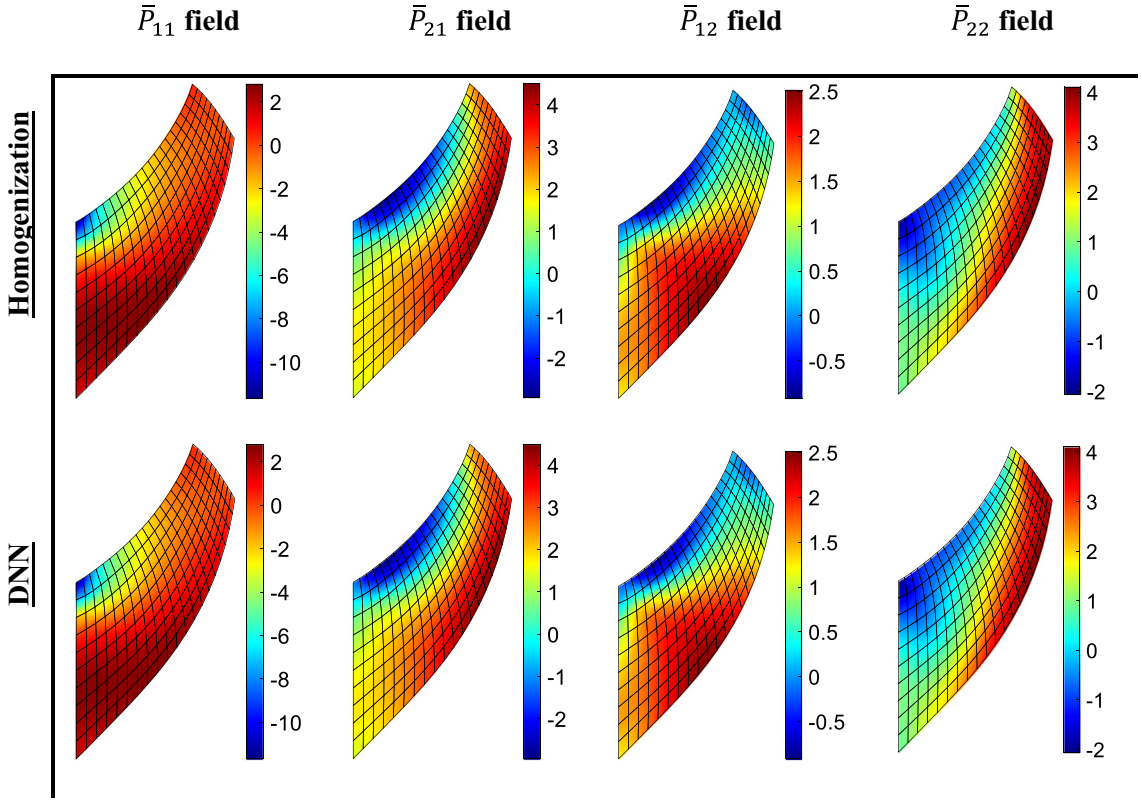


Fig. 27. Macroscopic 1st PK stress fields from DNN and homogenization based FE^2 analyses (no microstructure rotation).

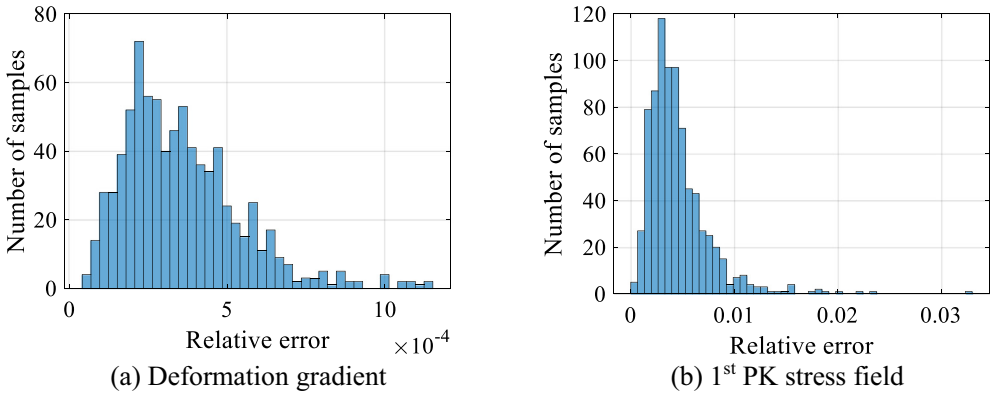


Fig. 28. Histograms of the relative errors of the macroscopic deformation gradient and 1st PK stress fields (no microstructure rotation).

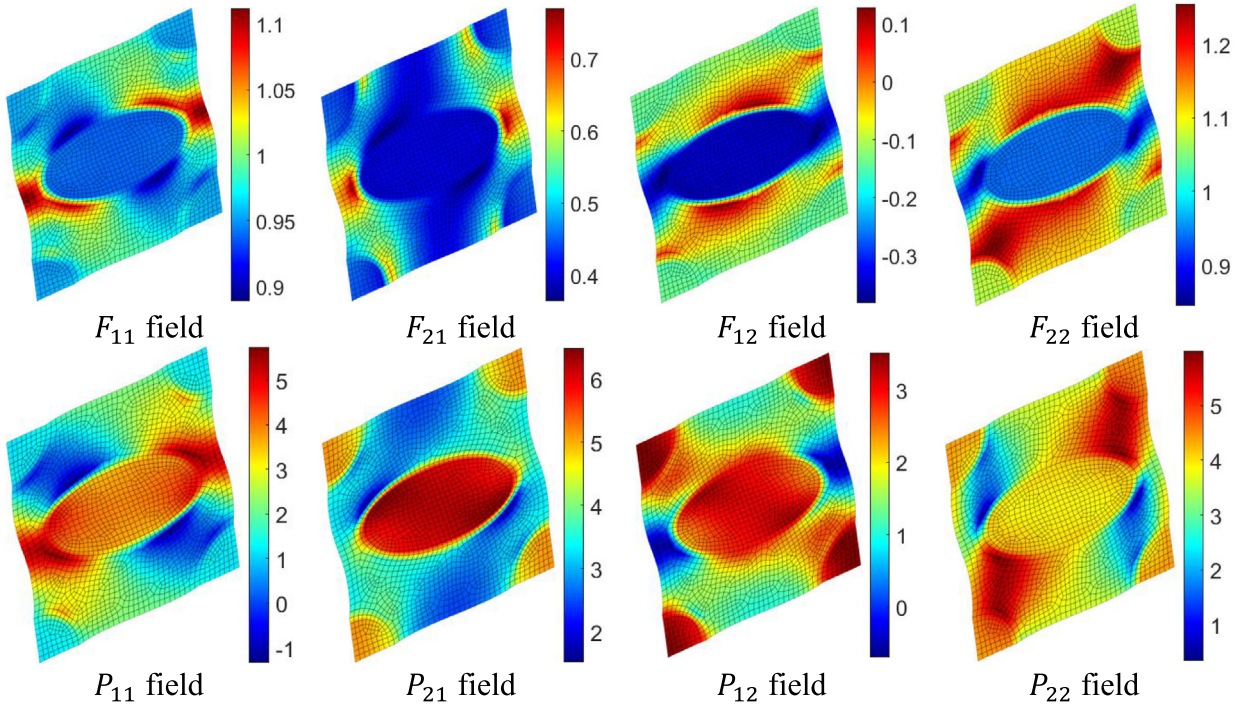


Fig. 29. Microscopic deformation gradient and 1st PK stress fields from homogenization analysis at macroscale integration point A in Fig. 24 (no microstructure rotation) with $\bar{\boldsymbol{F}} = [0.9713 \quad 0.4663 \quad -0.1611 \quad 1.0713]^T$.

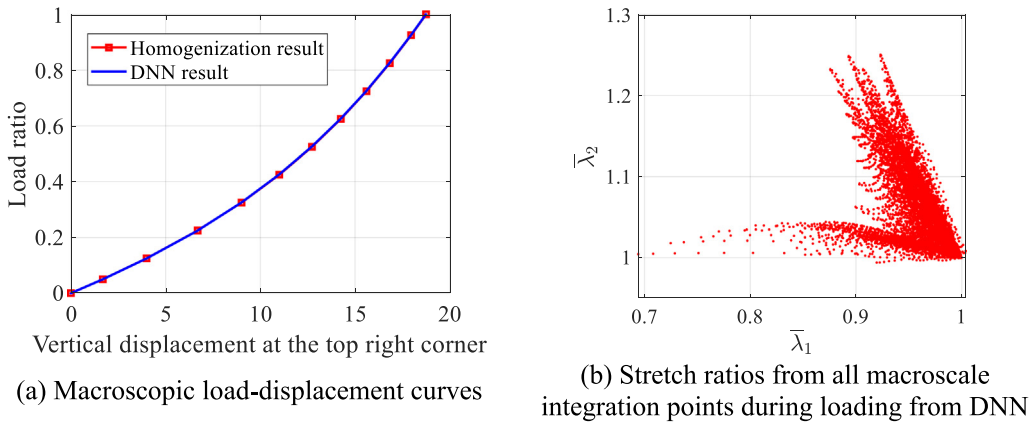


Fig. 30. Results of Cook's membrane with DNN and homogenization analysis on unit cell-1 (90° microstructure rotation).

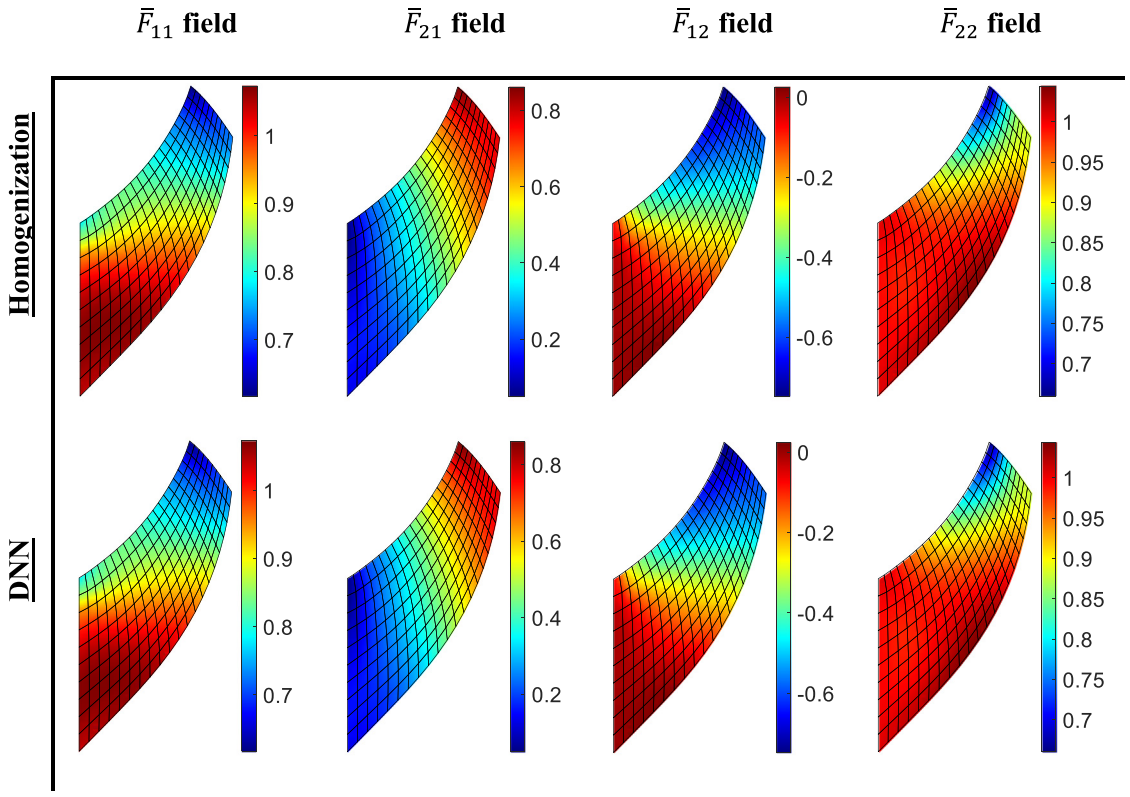


Fig. 31. Macroscopic deformation gradient fields from DNN and homogenization based FE^2 analyses of the Cook's membrane (90° microstructure rotation).

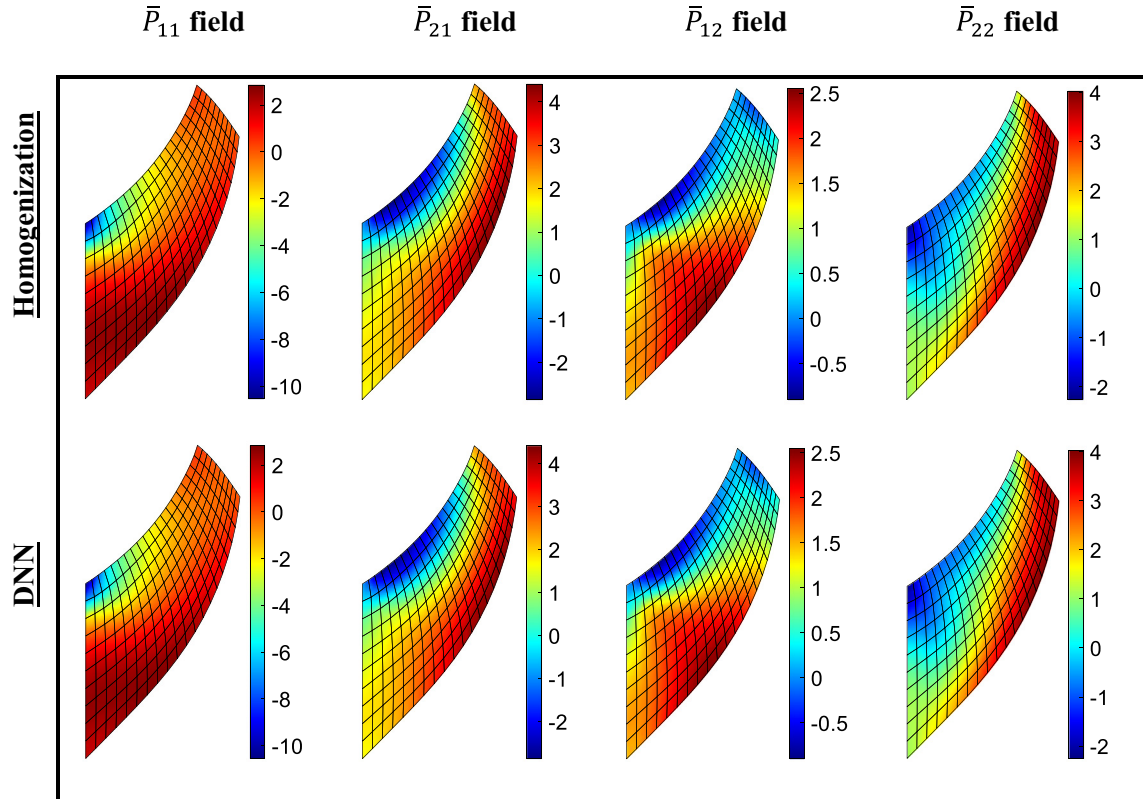


Fig. 32. Macroscopic 1st PK stress fields from DNN and homogenization based FE^2 analyses of the Cook's membrane (90° microstructure rotation).

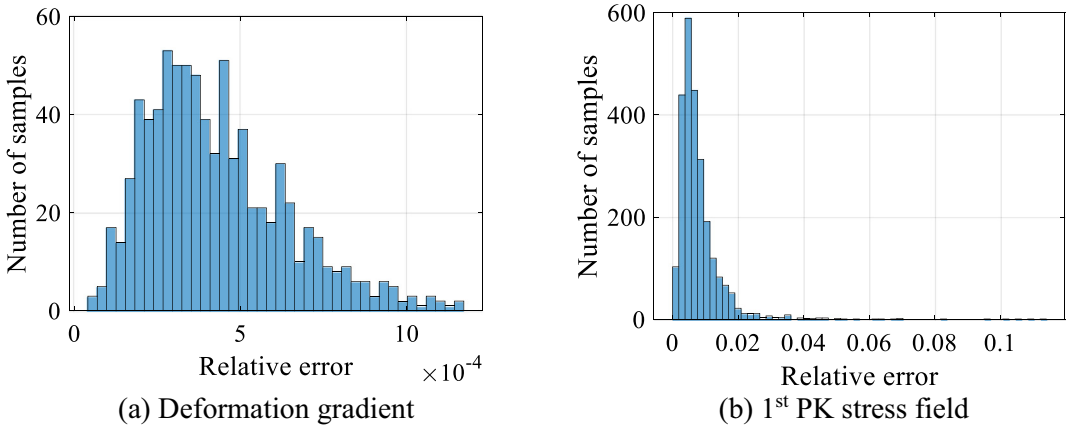


Fig. 33. Histograms of the relative errors of the macroscopic deformation gradient and 1st PK stress fields (90° microstructure rotation).

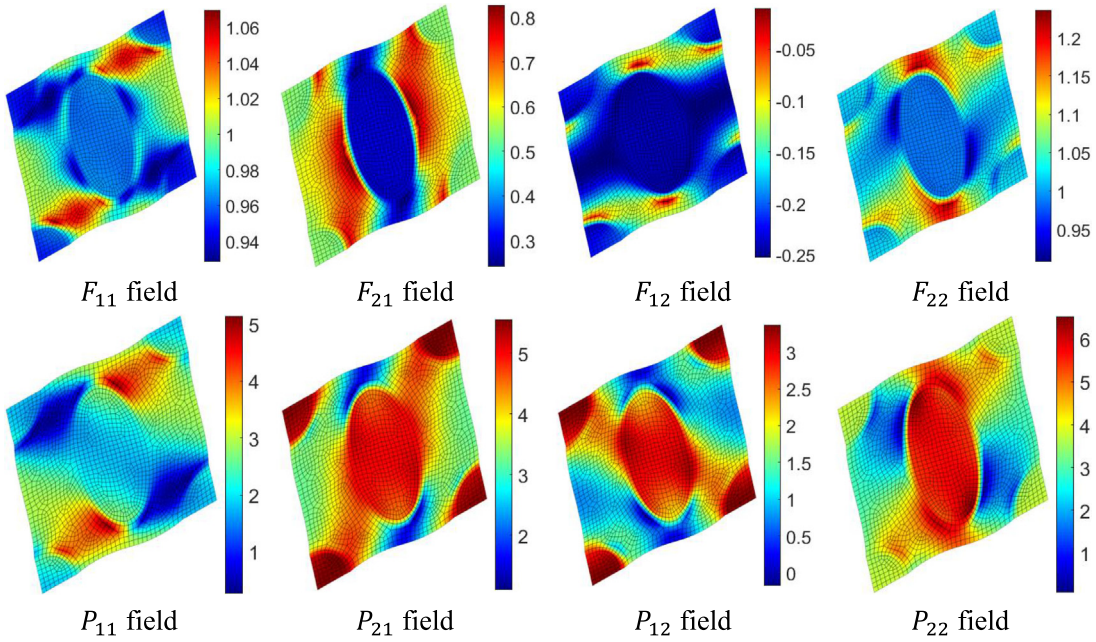


Fig. 34. Microscopic deformation gradient and 1st PK stress fields from homogenization analysis at macroscale integration point A in Fig. 24 (90° microstructure rotation) with $\bar{\boldsymbol{F}} = [0.9780 \quad 0.5253 \quad -0.2044 \quad 1.0331]^T$.

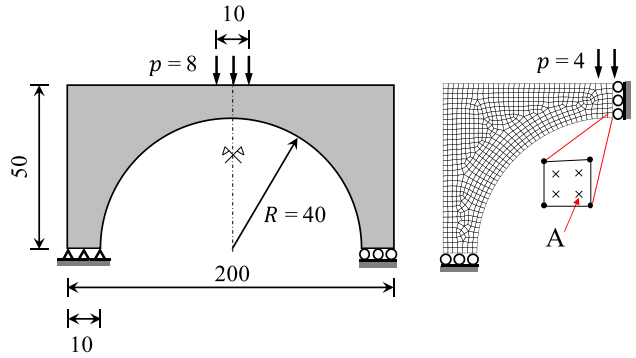


Fig. 35. Geometry and FE mesh of the (macroscale) arch problem.

loading step from homogenization analysis, where highly non-homogeneous distributions can be seen. This result shows that the trained DNN surrogate model can learn the complex behavior of the unit cell directly from the data.

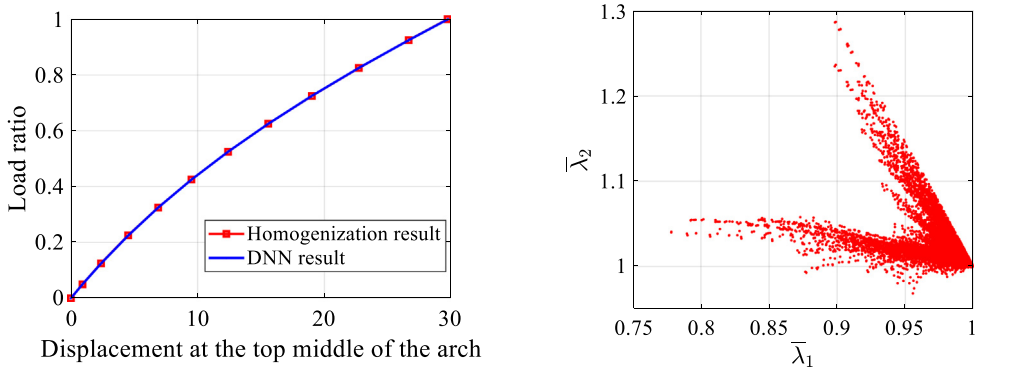
5.1.2. Rotation of microstructure by 90°

As the macroscopic material behavior of the media made of unit cell-1 in Fig. 4 is anisotropic, the orientation of the macroscopic material in applications can be different due to specific engineering considerations. This change of orientation of microstructure can be made outside the homogenization analysis and in the DNN surrogate model by considering a change of basis. For example, suppose the microstructure is rotated by an angle α (2D case) and a macroscopic deformation gradient $\bar{\boldsymbol{F}}$ is imposed, the deformation pattern of the rotated unit cell can be equivalently captured by the unrotated unit cell undergoing a macroscopic deformation gradient $\bar{\boldsymbol{F}}^*$ with

$$\bar{\boldsymbol{F}}^* = \boldsymbol{Q}(-\alpha).\bar{\boldsymbol{F}}.\boldsymbol{Q}(-\alpha)^T \tag{57}$$

where the rotation matrix $\boldsymbol{Q}$ is defined in Eq. $(40)_3$ and note that $\boldsymbol{Q}(-\alpha) = \boldsymbol{Q}(\alpha)^T$. The homogenized 1st PK stress tensor $\bar{\boldsymbol{P}}$ of the rotated unit cell under $\bar{\boldsymbol{F}}$ is then obtained by

$$\bar{\boldsymbol{P}} = \boldsymbol{Q}(\alpha).\bar{\boldsymbol{P}}^*.\boldsymbol{Q}(\alpha)^T \tag{58}$$



(a) Macroscopic load-displacement curves

(b) Stretch ratios from all macroscale integration points during loading from DNN

Fig. 36. Results of the arch problem with DNN and homogenization analysis on unit cell-2.

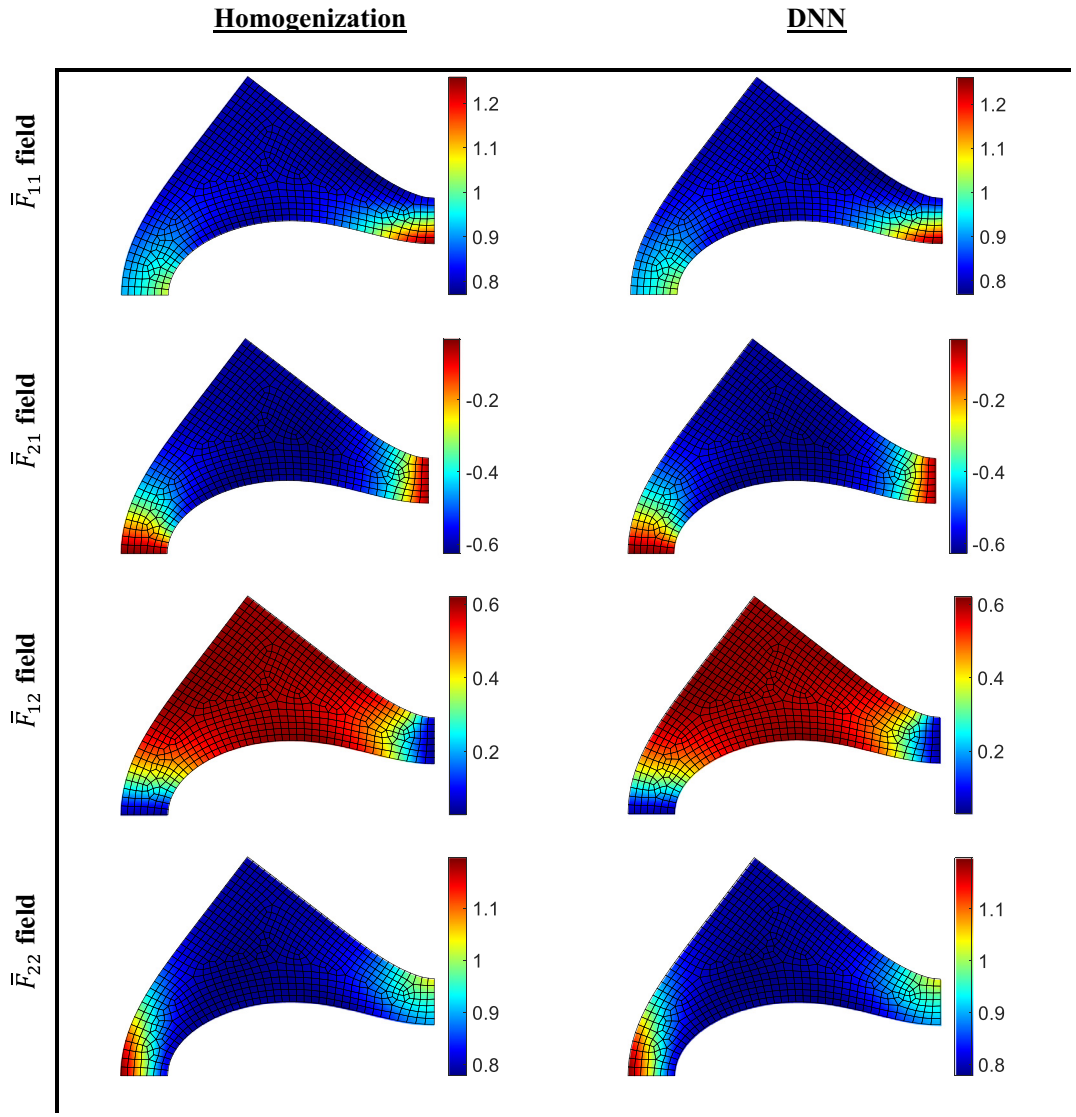


Fig. 37. Macroscopic deformation gradient fields from DNN and homogenization based FE^2 analyses of the arch problem.

where $\bar{\boldsymbol{P}}^*$ represents the homogenized 1st PK stress of the unrotated unit cell under $\bar{\boldsymbol{F}}^*$.

To show the idea of rotated unit cell microstructure, $\alpha = 90°$ is considered and a direct homogenization analysis on the rotated unit cell is carried out. Meanwhile, the FE^2 analysis with the trained DNN using the strategy in Eqs. (57) and (58) is also carried out. The two results are compared in Figs. 30–34, similarly as in Section 5.1.1, and Fig. 33 shows the corresponding relative errors

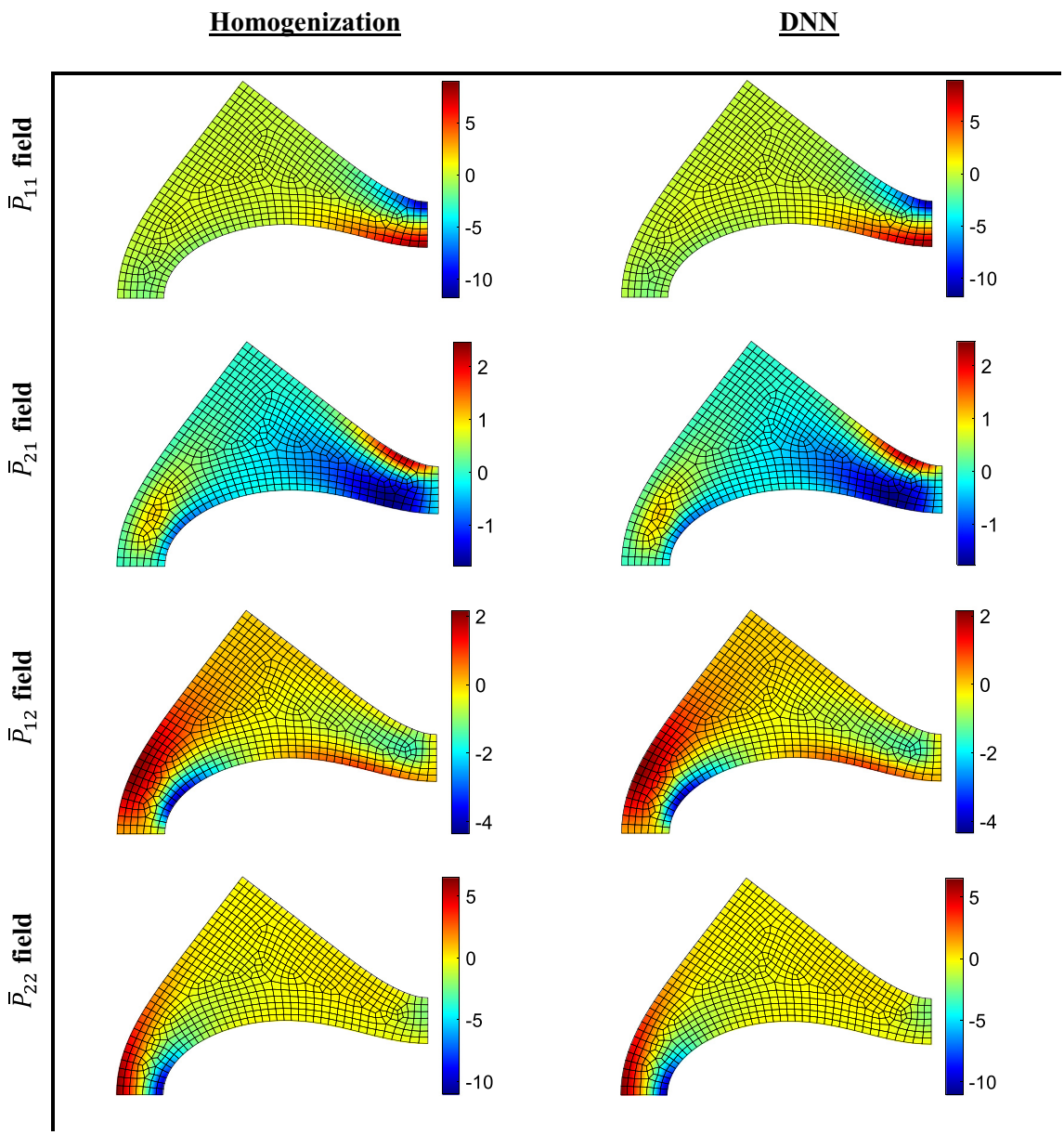


Fig. 38. Macroscopic 1st PK stress fields from DNN and homogenization based FE^2 analyses of the arch problem.

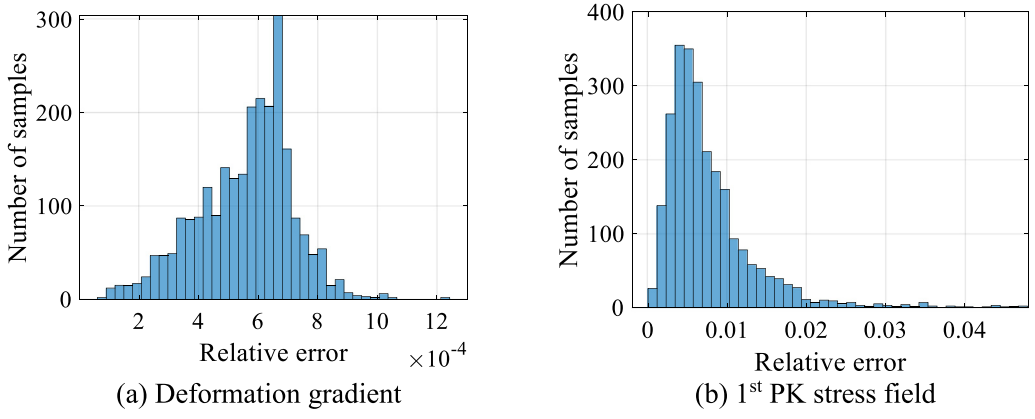


Fig. 39. Histograms of the relative errors of the macroscopic deformation gradient and 1st PK stress fields.

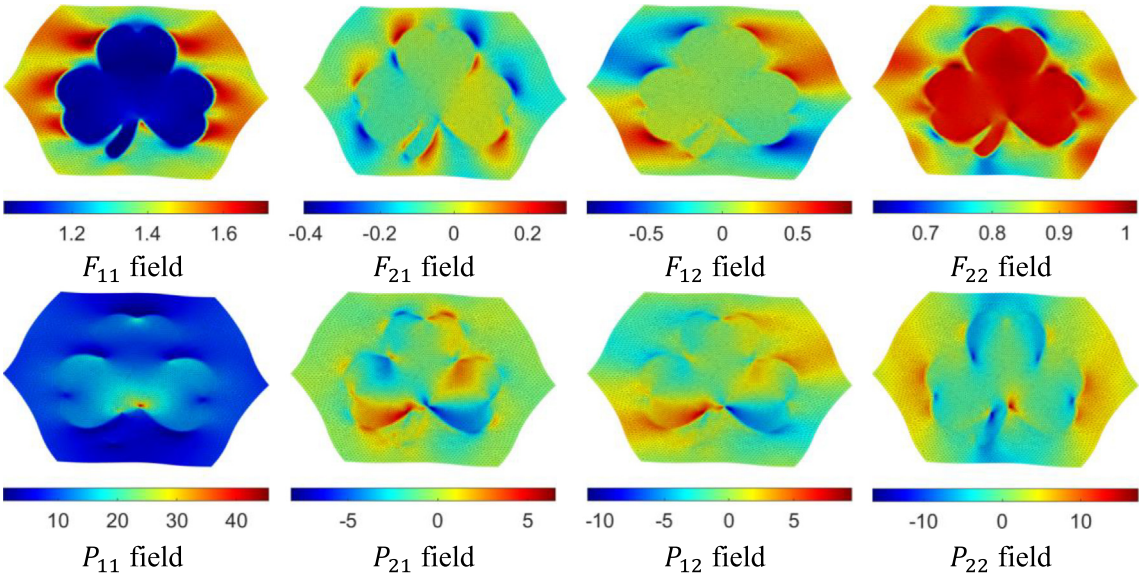


Fig. 40. Microscopic deformation gradient and 1st PK stress fields from homogenization analysis at macroscale integration point A in Fig. 35 with $\bar{\boldsymbol{F}} = [1.2851 \quad -0.0532 \quad 0.0191 \quad 0.8993]^T$.

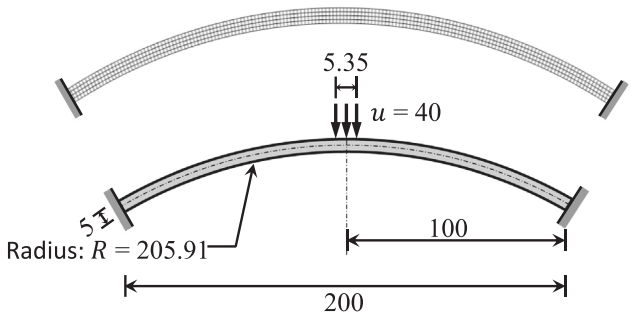
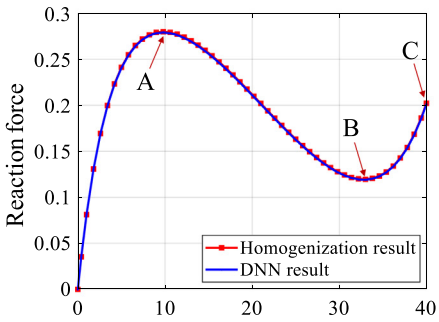


Fig. 41. Geometry and FE mesh of the (macroscale) shallow arch problem.

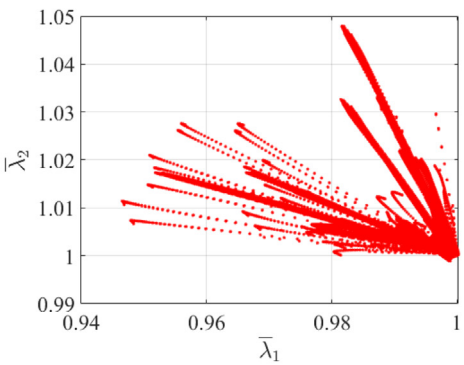
of the macroscopic deformation gradient and 1st PK stress fields from DNN based FE^2 analyses. The relative errors of the macro deformation gradient are bounded by 0.12% which happens at $\bar{\boldsymbol{F}} = [0.7159 \quad 0.6876 \quad -0.6406 \quad 0.7824]^T$, while the relative errors of the macro 1st PK stress are bounded by 3.53% that happens at $\bar{\boldsymbol{P}} = [-0.0703 \quad -0.0557 \quad -0.1481 \quad 0.1754]^T$. From the results, it can be seen that the trained DNN model, as combined with the transformation strategy in Eqs. (57) and (58), can be efficiently used to serve as the macroscale material subroutine irrespective of the orientation of the underlying microstructure.

5.2. Arch with unit cell-2

This example considers an arch-like structure with load applied at the center of its top surface as shown in Fig. 35, where using symmetry only half macro-domain is considered in FE^2 by assuming that there is no asymmetric bifurcation during the loading process. The macroscopic material is assumed to be periodic with unit cell in Fig. 5. Fig. 36a shows the macroscopic displacement-load curves from DNN surrogate and direct homogenization analysis in FE^2, where a good match can be seen. The stretch ratios at all macroscale integration points during the loading process from DNN based FE^2 analysis in Fig. 36b justifies the use of DNN. Figs. 37 and 38 plot the macroscale deformation gradient field and 1st PK stress field from the DNN based and homogenization based FE^2 analyses, where no distinct differences can be observed. Moreover, the histograms of relative errors of the two fields from the DNN based analyses are shown in Fig. 39, where the two error bounds are 0.12% and 11.39%, respectively, for the deformation



(a) Macroscopic load-displacement curves



(b) Stretch ratios from all macroscale integration points during loading from DNN

Fig. 42. Results of the shallow arch problem with DNN and homogenization analysis on unit cell-1.

gradient and 1st PK stress. The error bounds correspond to the points with $\bar{\boldsymbol{F}} = [0.7790 \quad -0.3009 \quad 0.3605 \quad 0.9863]^T$ and $\bar{\boldsymbol{P}} = [0.0023 \quad -0.0138 \quad -0.0094 \quad 0.0035]^T$. Finally, Fig. 40 shows the microscopic deformation gradient and 1st PK stress fields at the macroscale integration point $\boldsymbol{A}$ in Fig. 35 at the last loading step from homogenization analysis, where highly non-homogeneous deformation can be seen. These results again validate the efficacy of the trained DNN surrogate for reproducing results in FE^2 with high accuracy.

5.3. Shallow arch with unit cell-1

In this example, a shallow arch structure with displacement applied at the middle of the top surface is considered, as shown in Fig. 41. The macroscopic material consists of periodic unit cells shown in Fig. 4. The displacement-load curves from DNN and homogenization based FE^2 analyses are plotted in Fig. 42a which show good matches in snap-through behaviors for this example. Fig. 42b shows the stretch ratios at all macroscopic integration points during the loading process, which confirms that the DNN model works well within the training data range. The deformed shapes at loading states A, B, and C in Fig. 42a from both DNN and homogenization based FE^2 analyses are compared in Fig. 43, where good matches can be observed. Furthermore, Fig. 44, Fig. 45, and Fig. 46 plot the relative error statistics of macroscopic deformation gradient and 1st PK stress fields from DNN based FE^2 analysis at loading states A, B, and C, respectively. At loading state A, the maximum relative error is 0.043% at the deformation gradient $\bar{\boldsymbol{F}} = [1.0039 \quad 0.0510 \quad 0.0615 \quad 0.9960]^T$ and 6.78% for 1st PK stress $\bar{\boldsymbol{P}} = [-0.0015 \quad 0.0037 \quad 0.0026 \quad 0.0234]^T$. At loading state B, the relative error is bounded by 0.075% for the deformation gradient $\bar{\boldsymbol{F}} = [0.8794 \quad 0.4341 \quad -0.4400 \quad 0.9087]^T$ and 9.28% for 1st PK stress $\bar{\boldsymbol{P}} = [0.0103 \quad 0.0343 \quad 0.0172 \quad 0.0175]^T$. At loading state C, the maximum relative errors are 0.061% for deformation gradient $\bar{\boldsymbol{F}} = [0.8746 \quad -0.4427 \quad 0.4500 \quad 0.9043]^T$ and 9.87% for 1st PK stress $\bar{\boldsymbol{P}} = [0.0273 \quad -0.0269 \quad -0.02700.0153]^T$. All these results show the efficacy of the trained DNN model in the FE^2 analysis for structures with complex mechanical behaviors.

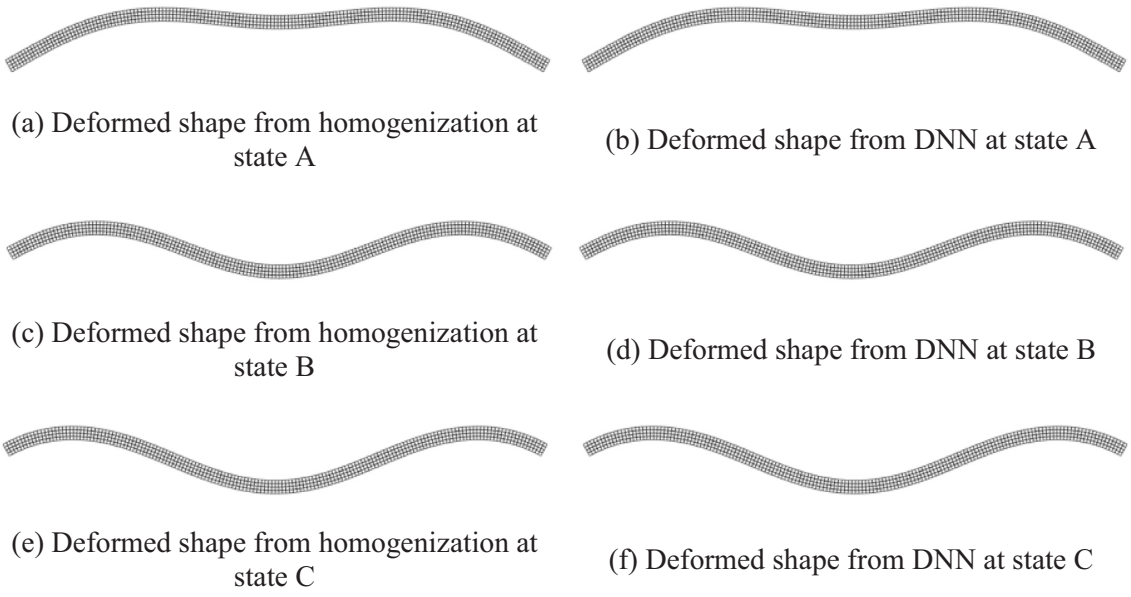


Fig. 43. Deformed shapes from homogenization (*left column*) and DNN (*right column*) results at certain states.

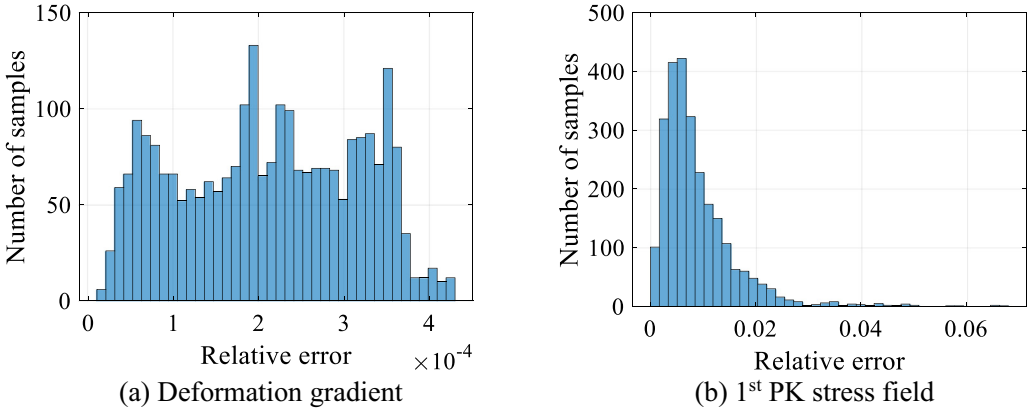


Fig. 44. Histograms of the relative errors of the macroscopic deformation gradient and 1st PK stress fields at loading state A.

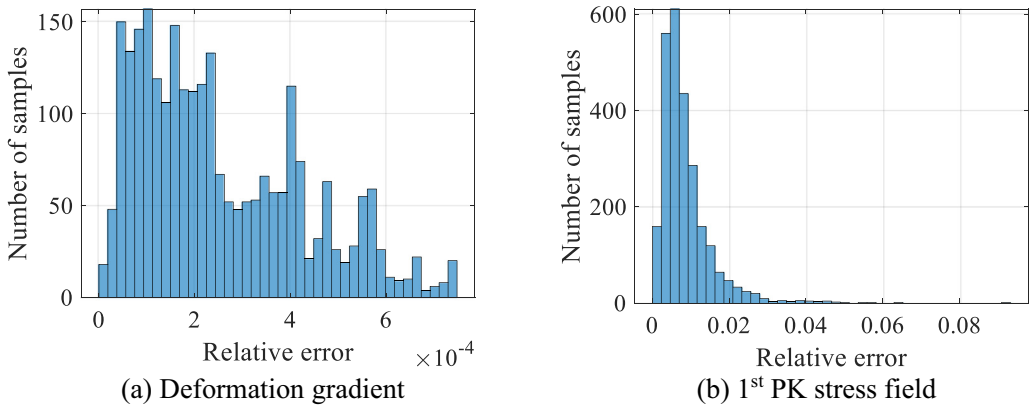


Fig. 45. Histograms of the relative errors of the macroscopic deformation gradient and 1st PK stress fields at loading state B.

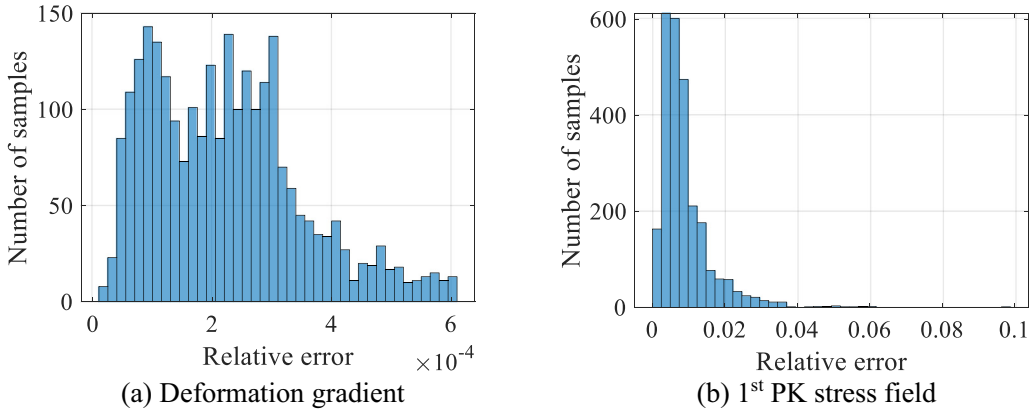


Fig. 46. Histograms of the relative errors of the macroscopic deformation gradient and 1st PK stress fields at loading state C.

6. Conclusions

In this paper, the use of a data-driven approach is explored to expedite the computationally expensive finite strain FE^2 analysis. In particular, the homogenization analysis at each macro integration point is replaced by a trained DNN surrogate model, which obviates the need for finite element analysis on the underlying unit cell. To fulfill the requirement of frame indifference, the rigid-body rotation part is separated from the deformation gradient, and only the work conjugate pair $(\bar{\boldsymbol{C}}, \bar{\boldsymbol{S}})$ is used in the surrogate model, and the push-forward operations are then applied outside the surrogate material subroutine. The surrogate modeling falls into the category of a regression problem, where the mapping $\mathcal{M}: \bar{c} \rightarrow \bar{s}$ has to be learned from data. Formulated in a consistent probabilistic framework, the regression task by machine learning with deep neural networks is presented from a theoretical viewpoint. In preparation of training data, an efficient radial sampling strategy from the macroscopic deformation space is proposed, where samples are grouped into different loading paths and are generated with a smaller number of homogenization analyses. Different neural network architectures (i.e., fully connected layers, convolutional layers, and ResNet connections) and training strategies (regular and Sobolev), are investigated. It is shown that Sobolev training, with additional derivative information, leads to higher training accuracy as compared to regular training, and a combination of different types of layers and connections gives an improved surrogate model performance. The high accuracy of the DNN surrogates is demonstrated on uniformly distributed testing samples in the considered training dataspace. The DNN surrogate is shown to lead to a significant improvement in the computational expense as compared to direct homogenization. Finally, a series of FE^2 examples are examined, where a close match between the direct FE^2 analysis with homogenization and FE^2 with DNN surrogate is observed. The results presented in this paper show the clear potential of DNN in regression tasks that are encountered in computationally expensive multiscale FE^2 analysis. Future work will focus on 3D metamaterials and inelastic material behaviors.

Declaration of Competing Interest

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

Acknowledgments

The presented work is supported in part by the USNational Science Foundation through grant CMMI-1762277. Any opinions, findings, conclusions, and recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the sponsors.

Appendix A. Implementation details: FE^2 Analysis

This appendix gives details on the implementation of the FE^2 analysis. The implementation details on the homogenization analysis in Section A.2 has been presented in our previous paper [56] and is given again here for the completeness. Some derivation pro-

cess is omitted for the sake of brevity, and interested readers are referred to Ref. [56].

A.1. Macroscale analysis

The macroscale BVP is solved using standard Galerkin based finite element analysis. The finite element discretized weak form of Eq. (1) is

$$\bar{\boldsymbol{R}} = \bar{\boldsymbol{F}}_{int} - \bar{\boldsymbol{F}}_{ext} = \boldsymbol{0} \tag{A1}$$

with

$$\bar{\boldsymbol{F}}_{int} = \mathop{\mathcal{A}}_{e=1}^{\bar{n}_{ele}} \bar{\boldsymbol{F}}^e_{int} \text{ with } \bar{\boldsymbol{F}}^e_{int} = \int_{\bar{\Omega}^e_0} \boldsymbol{B}^T \bar{\boldsymbol{P}} dV$$

$$\bar{\boldsymbol{F}}_{ext} = \int_{\partial\bar{\Omega}_{0\sigma}} \boldsymbol{N}^T \bar{\boldsymbol{T}} dA \tag{A2}$$

where $\boldsymbol{N}$ and $\boldsymbol{B}$ represent the shape function matrix and its derivatives such that $\bar{\boldsymbol{u}} = \boldsymbol{N}\bar{\boldsymbol{u}}^e$ and $\bar{\boldsymbol{F}} = \boldsymbol{B}\bar{\boldsymbol{u}}^e$ with $\bar{\boldsymbol{u}}^e$ the element displacement vector and $\bar{\boldsymbol{u}}$ and $\bar{\boldsymbol{F}}$ the vector form of displacement and deformation gradient. Here $\bar{\boldsymbol{P}}$ and $\bar{\boldsymbol{T}}$ are the vector forms of 1st PK stress and applied PK traction. The e^{th} element integration domain is denoted by $\bar{\Omega}^e_0$ such that $\bar{\Omega}_0 = \bigcup_{e=1}^{\bar{n}_{ele}} \bar{\Omega}^e_0$ where $\bar{n}_{ele}$ is the number of element. The material subroutine is given by the homogenization analysis in the next section. To ensure asymptotic quadratic convergence rate of Newton-Raphson (NR) solver, the structural stiffness matrix is calculated by

$$\bar{\boldsymbol{K}}_T = \frac{\partial \bar{\boldsymbol{R}}}{\partial \bar{\boldsymbol{u}}} = \mathop{\mathcal{A}}_{e=1}^{n_{ele}} \bar{\boldsymbol{k}}^e_T \text{ with } \bar{\boldsymbol{k}}^e_T = \int_{\bar{\Omega}^e_0} \boldsymbol{B}^T [\bar{\mathbb{A}}] \boldsymbol{B} dV \tag{A3}$$

where $[\bar{\mathbb{A}}]$ is a matrix form of the 4th-oder tensor $\bar{\mathbb{A}}$.

A.2. Microscale (homogenization) analysis

In the deformation driven scheme, the macroscopic deformation gradient $\bar{\boldsymbol{F}}$ is given at the current macroscale integration point. The 1st-order homogenization is then driven by $\bar{\boldsymbol{F}}$. The periodic boundary condition in Eq. (9) together with the rigid-body constraint are enforced through Lagrange multipliers. After finite element discretization, the nonlinear system of equations reads

$$\boldsymbol{R}(\boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \begin{bmatrix} \boldsymbol{R}_1(\boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \\ \boldsymbol{R}_2(\boldsymbol{u}) \\ \boldsymbol{R}_3(\boldsymbol{u}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_{int}(\boldsymbol{u}) - \boldsymbol{A}_1^T\boldsymbol{\lambda} - \boldsymbol{A}_2^T\boldsymbol{\mu} \\ -\boldsymbol{A}_1\boldsymbol{u} \\ -\boldsymbol{A}_2\boldsymbol{u} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{h} \end{bmatrix} = \boldsymbol{0} \tag{A4}$$

where $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ represent Lagrange multiplier vectors that are used to enforce rigid-body translation constraint and periodic boundary conditions, respectively. $\boldsymbol{F}_{int}$ represents the global internal force vector defined by

$$\boldsymbol{F}_{int}(\boldsymbol{u}) = \mathop{\mathcal{A}}_{e=1}^{n_{ele}} \boldsymbol{F}^e_{int} \text{ with } \boldsymbol{F}^e_{int} = \int_{\Omega^e_0} \boldsymbol{B}^T \boldsymbol{P} dV \tag{A5}$$

where Ω^e_0 represents the e^{th} element integration domain satisfying $\mathcal{B}_0 = \bigcup_{e=1}^{n_{ele}} \Omega^e_0$ and n_{ele} are the total number of elements in the RVE. It is remarked here that $\boldsymbol{B}$ has the same meaning as that in Section A.1 but is pertaining to the finite element mesh of RVE rather than the macroscale BVP finite element mesh.

The matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$, and vector $\boldsymbol{h}$ are constructed such that (for 2D problem)

$$\boldsymbol{u}_o = \boldsymbol{A}_1\boldsymbol{u}$$

$$\boldsymbol{u}^+ - \boldsymbol{u}^- = \boldsymbol{A}_2\boldsymbol{u}$$

$$\boldsymbol{h} = \begin{bmatrix} (\bar{\boldsymbol{F}} - \boldsymbol{I}).\boldsymbol{L}_1 \\ \vdots \\ (\bar{\boldsymbol{F}} - \boldsymbol{I}).\boldsymbol{L}_m \end{bmatrix} = [\boldsymbol{L}_M]\left([\bar{\boldsymbol{F}}] - [\boldsymbol{I}]\right)$$

$$= \begin{bmatrix} \tilde{X}_1 & 0 & \tilde{Y}_1 & 0 \\ 0 & \tilde{X}_1 & 0 & \tilde{Y}_1 \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{X}_m & 0 & \tilde{Y}_m & 0 \\ 0 & \tilde{X}_m & 0 & \tilde{Y}_m \end{bmatrix}_{2m\times 4} \left(\begin{bmatrix} \bar{F}_{11} \\ \bar{F}_{21} \\ \bar{F}_{12} \\ \bar{F}_{22} \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right) \tag{A6}$$

where $\boldsymbol{u}$ is the global nodal displacement vector, $\boldsymbol{u}^+ = \left[\boldsymbol{u}^+_1, \cdots, \boldsymbol{u}^+_m\right]^T$ and $\boldsymbol{u}^- = \left[\boldsymbol{u}^-_1, \cdots, \boldsymbol{u}^-_m\right]^T$ includes m nodal displacements defined on the positive and negative boundary sides, respectively. $\boldsymbol{L}_q = \left[\tilde{X}_q, \tilde{Y}_q\right]^T$ is the translational vector from the q^{th} node on the negative side to the q^{th} node on the positive side. Here $\boldsymbol{u}_o$ is the displacement of an arbitrarily chosen node that is fixed to remove rigid-body translation.

The nonlinear system in Eq. (A4) is solved using the NR method and the Jacobian matrix, which is needed for NR solver, can be calculated as

$$[\boldsymbol{J}_T] = \begin{bmatrix} \partial\boldsymbol{R}_1/\partial\boldsymbol{u} & \partial\boldsymbol{R}_1/\partial\boldsymbol{\lambda} & \partial\boldsymbol{R}_1/\partial\boldsymbol{\mu} \\ \partial\boldsymbol{R}_2/\partial\boldsymbol{u} & \partial\boldsymbol{R}_2/\partial\boldsymbol{\lambda} & \partial\boldsymbol{R}_2/\partial\boldsymbol{\mu} \\ \partial\boldsymbol{R}_3/\partial\boldsymbol{u} & \partial\boldsymbol{R}_3/\partial\boldsymbol{\lambda} & \partial\boldsymbol{R}_3/\partial\boldsymbol{\mu} \end{bmatrix} = \begin{bmatrix} \boldsymbol{K}_T & -\boldsymbol{A}_1^T & -\boldsymbol{A}_2^T \\ -\boldsymbol{A}_1 & \boldsymbol{0} & \boldsymbol{0} \\ -\boldsymbol{A}_2 & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \tag{A7}$$

where the term $\boldsymbol{K}_T$ is the tangent structural stiffness matrix calculated by

$$\boldsymbol{K}_T = \frac{\partial \boldsymbol{F}_{int}}{\partial \boldsymbol{u}} = \mathop{\mathcal{A}}_{e=1}^{n_{ele}} \boldsymbol{k}^e_T \text{ with } \boldsymbol{k}^e_T = \int_{\Omega^e} \boldsymbol{B}^T [\mathbb{A}] \boldsymbol{B} dV \text{ and } \mathbb{A} = \frac{\partial \boldsymbol{P}}{\partial \boldsymbol{F}} \tag{A8}$$

in which the tangent moduli $\mathbb{A}$ is obtained from material subroutine. In this study, a regularized neo-Hookean hyperelasticity model is adopted to simulate the constitutive behaviors of different underlying materials in the microstructure, of which the free energy reads

$$\psi(\boldsymbol{C}) = \frac{1}{2}\kappa(J-1)^2 + \frac{\mu}{2}\left(\hat{I}_1 - 3\right) \tag{A9}$$

where $\boldsymbol{C}$ is the right Cauchy-Green tensor and $\hat{I}_1$ is the first invariant of $\hat{\boldsymbol{C}}$, i.e., $\hat{I}_1 = \text{tr}\hat{\boldsymbol{C}}$ with $\hat{\boldsymbol{C}} = J^{-2/3}\boldsymbol{C}$ and J the determinant of the deformation gradient. κ and μ are bulk and shear modulus of the material.

A.2.1. Homogenized stress and tangent moduli

Using Eq. $(11)_2$ and the definition of matrix $[\boldsymbol{L}_M]$ given in Eq. (A6), the homogenized stress $\bar{\boldsymbol{P}}$ is computed as

$$[\bar{\boldsymbol{P}}] = \frac{1}{V}[\boldsymbol{L}_M]^{\text{T}}\boldsymbol{\mu} \tag{A10}$$

where the bracket outside $\bar{\boldsymbol{P}}$ means that it is arranged in a 4×1 vector form (2D), similarly as $[\bar{\boldsymbol{F}}]$ used in Eq. (A6).

The 4th-order homogenized tangent moduli $\bar{\mathbb{A}}$ is rephrased in a matrix form as $[\bar{\mathbb{A}}] = \partial[\bar{\boldsymbol{P}}]/\partial[\bar{\boldsymbol{F}}]$. From Eq. (A10), it is clear that $[\bar{\mathbb{A}}]$ is determined by the derivative of Lagrange multiplier $\boldsymbol{\mu}$ with respect to $\bar{\boldsymbol{F}}$. To this end, the set of global equilibrium equation (Eq. (A4)) is perturbed at the equilibrium state by a perturbation $\Delta\bar{\boldsymbol{F}}$ and by a straightforward manipulation it finally gives

$$[\bar{\mathbb{A}}] = -\frac{1}{V}\left[\hat{\boldsymbol{L}}_M\right]^T [\boldsymbol{J}_T]^{-1} \left[\hat{\boldsymbol{L}}_M\right] \tag{A11}$$

where the matrix $\left[\hat{\boldsymbol{L}}_M\right]$ is of size $(N + 2 + 2m) \times 4$ for a 2D case and is defined by

$$\left[\hat{\boldsymbol{L}}_M\right] = \begin{bmatrix} \mathbf{0}_{N\times 4} \\ \mathbf{0}_{2\times 4} \\ [\boldsymbol{L}_M]_{2m\times 4} \end{bmatrix} \tag{A12}$$

in which N represents the total number of DOFs in the displacement field $\boldsymbol{u}$, i.e., the size of $\boldsymbol{u}$ vector.

References

[1] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556; 2014.
[2] Russakovsky O et al. ImageNet Large Scale Visual Recognition Challenge. Int J Comput Vision 2015;115(3):211–52. https://doi.org/10.1007/s11263-015-0816-y.
[3] Szegedy C et al. Going deeper with convolutions. Proceedings of the IEEE conference on computer vision and pattern recognition, 2015.
[4] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature 1986;323(6088):533–6.
[5] LeCun Y et al. Gradient-based learning applied to document recognition. Proc IEEE 1998;86(11):2278–324.
[6] LeCun Y et al. Backpropagation applied to handwritten zip code recognition. Neural Comput 1989;1(4):541–51.
[7] Le Cun Y et al. Handwritten digit recognition with a back-propagation network. Proceedings of the 2nd International Conference on Neural Information Processing Systems, 1989.
[8] Gers FA, Schmidhuber J, Cummins F. Learning to forget: Continual prediction with LSTM; 1999.
[9] Gers FA, Schraudolph NN, Schmidhuber J. Learning precise timing with LSTM recurrent networks. J Mach Learn Res 2002;3(Aug):115–43.
[10] Gers FA, Schmidhuber J, Cummins F. Learning to Forget: Continual Prediction with LSTM. Neural Comput 2000;12(10):2451–71. https://doi.org/10.1162/089976600300015015.
[11] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 2012.
[12] He K et al. Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016.
[13] Huang G et al. Densely connected convolutional networks. Proceedings of the IEEE conference on computer vision and pattern recognition, 2017.
[14] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. Proceedings of the thirteenth international conference on artificial intelligence and statistics, 2010.
[15] He K et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. Proceedings of the IEEE international conference on computer vision, 2015.
[16] Sussillo D, Abbott L. Random walk initialization for training very deep feedforward networks. arXiv preprint arXiv:1412.6558; 2014.
[17] Mishkin D, Matas J. All you need is a good init. arXiv preprint arXiv:1511.06422; 2015.
[18] Ioffe S, Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: Francis B, David B, editors. Proceedings of the 32nd International Conference on Machine Learning. PMLR: Proceedings of Machine Learning Research; 2015. p. 448–56.
[19] Srivastava N et al. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 2014;15(1):1929–58.
[20] Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980; 2014.
[21] Zeiler MD, Fergus R. Visualizing and understanding convolutional networks. European conference on computer vision. Springer; 2014.
[22] Graves A, Jaitly N. Towards end-to-end speech recognition with recurrent neural networks. International conference on machine learning, 2014.
[23] Wu Y, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144; 2016.
[24] Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. Advances in neural information processing systems, 2014.
[25] Kim Y. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882; 2014.
[26] Conneau A, et al. Very deep convolutional networks for natural language processing. arXiv preprint arXiv:1606.01781, vol. 2; 2016.
[27] Bock FE et al. A review of the application of machine learning and data mining approaches in continuum materials mechanics. Front Mater 2019;6:110.
[28] Koeppe A, Bamer F, Markert B. An intelligent nonlinear meta element for elastoplastic continua: deep learning using a new Time-distributed Residual U-Net architecture. Comput Methods Appl Mech Eng 2020;366:. https://doi.org/10.1016/j.cma.2020.113088113088.
[29] Im S et al. Neural network constitutive model for crystal structures. Comput Mech 2021;67(1):185–206.
[30] Jokar M, Semperlotti F. Finite element network analysis: A machine learning based computational framework for the simulation of physical systems. Comput Struct 2021;247:. https://doi.org/10.1016/j.compstruc.2021.106484106484.
[31] Torky AA, Ohno S. Deep learning techniques for predicting nonlinear multi-component seismic responses of structural buildings. Comput Struct 2021;252:. https://doi.org/10.1016/j.compstruc.2021.106570106570.
[32] Ikumi T et al. Neural network-aided prediction of post-cracking tensile strength of fibre-reinforced concrete. Comput Struct 2021;256:. https://doi.org/10.1016/j.compstruc.2021.106640106640.
[33] Saeb S, Steinmann P, Javili A. Aspects of Computational Homogenization at Finite Deformations: A Unifying Review From Reuss' to Voigt's Bound. Appl Mech Rev 2016;68(5). https://doi.org/10.1115/1.4034024.
[34] Patel DK, Waas AM, Yen C-F. Direct numerical simulation of 3D woven textile composites subjected to tensile loading: An experimentally validated multiscale approach. Compos B Eng 2018;152:102–15. https://doi.org/10.1016/j.compositesb.2018.06.012.
[35] Tikarrouchine E et al. Non-linear FE2 multiscale simulation of damage, micro and macroscopic strains in polyamide 66-woven composite structures: Analysis and experimental validation. Compos Struct 2021;255:. https://doi.org/10.1016/j.compstruct.2020.112926112926.
[36] Karakoç A, Paltakari J, Taciroglu E. On the computational homogenization of three-dimensional fibrous materials. Compos Struct 2020;242:. https://doi.org/10.1016/j.compstruct.2020.112151112151.
[37] Rohan E, Heczko J. Homogenization and numerical modelling of poroelastic materials with self-contact in the microstructure. Comput Struct 2020;230:. https://doi.org/10.1016/j.compstruc.2019.06.003106086.
[38] Krejčí T et al. Effective elastic and fracture properties of regular and irregular masonry from nonlinear homogenization. Comput Struct 2021;254:. https://doi.org/10.1016/j.compstruc.2021.106580106580.
[39] Unger JF, Könke C. Coupling of scales in a multiscale simulation using neural networks. Comput Struct 2008;86(21):1994–2003. https://doi.org/10.1016/j.compstruc.2008.05.004.
[40] Unger JF, Könke C. Neural networks as material models within a multiscale approach. Comput Struct 2009;87(19–20):1177–86.
[41] Le BA, Yvonnet J, He Q-C. Computational homogenization of nonlinear elastic materials using neural networks. Int J Numer Meth Eng 2015;104(12):1061–84. https://doi.org/10.1002/nme.4953.
[42] Minh Nguyen-Thanh V et al. A surrogate model for computational homogenization of elastostatics at finite strain using high-dimensional model representation-based neural network. Int J Numer Meth Eng 2020;121(21):4811–42.
[43] Bessa MA et al. A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality. Comput Methods Appl Mech Eng 2017;320:633–67.
[44] Huang DZ et al. Learning constitutive relations from indirect observations using deep neural networks. J Comput Phys 2020;416:109491.
[45] Xu K, Huang DZ, Darve E. Learning constitutive relations using symmetric positive definite neural networks. J Comput Phys 2021;428:. https://doi.org/10.1016/j.jcp.2020.110072110072.
[46] Fernández M et al. Anisotropic hyperelastic constitutive models for finite deformations combining material theory and data-driven approaches with application to cubic lattice metamaterials. Comput Mech 2021;67(2):653–77. https://doi.org/10.1007/s00466-020-01954-7.
[47] Ghavamian F, Simone A. Accelerating multiscale finite element simulations of history-dependent materials using a recurrent neural network. Comput Methods Appl Mech Eng 2019;357:112594.
[48] Logarzo HJ, Capuano G, Rimoli JJ. Smart constitutive laws: Inelastic homogenization through machine learning. Comput Methods Appl Mech Eng 2021;373:113482.
[49] Wu L et al. A recurrent neural network-accelerated multi-scale model for elasto-plastic heterogeneous materials subjected to random cyclic and non-proportional loading paths. Comput Methods Appl Mech Eng 2020;369:. https://doi.org/10.1016/j.cma.2020.113234113234.
[50] Linka K et al. Constitutive artificial neural networks: A fast and general approach to predictive data-driven constitutive modeling by deep learning. J Comput Phys 2021;429:. https://doi.org/10.1016/j.jcp.2020.110010110010.
[51] Rao C, Liu Y. Three-dimensional convolutional neural network (3D-CNN) for heterogeneous material homogenization. Comput Mater Sci 2020;184:. https://doi.org/10.1016/j.commatsci.2020.109850109850.
[52] Xiao S et al. Machine learning in multiscale modeling of spatially tailored materials with microstructure uncertainties. Comput Struct 2021;249:. https://doi.org/10.1016/j.compstruc.2021.106511106511.
[53] Czarnecki WM, et al. *Sobolev training for neural networks.* arXiv preprint arXiv:1706.04859; 2017.
[54] Castañeda PP. Exact second-order estimates for the effective mechanical properties of nonlinear composite materials. J Mech Phys Solids 1996;44(6):827–62. https://doi.org/10.1016/0022-5096(96)00015-4.
[55] Shrimali B, Lefèvre V, Lopez-Pamies O. A simple explicit homogenization solution for the macroscopic elastic response of isotropic porous elastomers. J Mech Phys Solids 2019;122:364–80. https://doi.org/10.1016/j.jmps.2018.09.026.
[56] Zhang G, Feng N, Khandelwal K. A Computational framework for homogenization and multiscale stability analyses of nonlinear periodic materials. Int J Num Meth Eng 2021. https://doi.org/10.1002/nme.6802.
[57] Geymonat G, Müller S, Triantafyllidis N. Homogenization of nonlinearly elastic materials, microscopic bifurcation and macroscopic loss of rank-one convexity. Arch Ration Mech Anal 1993;122(3):231–90. https://doi.org/10.1007/BF00380256.

[58] Triantafyllidis N, Nestorović MD, Schraad MW. Failure Surfaces for Finitely Strained Two-Phase Periodic Solids Under General In-Plane Loading. J Appl Mech 2005;73(3):505–15. https://doi.org/10.1115/1.2126695.
[59] de Souza Neto EA et al. An RVE-based multiscale theory of solids with micro-scale inertia and body force effects. Mech Mater 2015;80:136–44. https://doi.org/10.1016/j.mechmat.2014.10.007.
[60] Hill R. On constitutive macro-variables for heterogeneous solids at finite strain. Proc Roy Soc Lond A Math Phys Sci 1972;326(1565):131–47. https://doi.org/10.1098/rspa.1972.0001.
[61] Mandel J. Plasticité classique et viscoplasticité (CISM Lecture notes, Udine, Italy). Vienna: Springer-Verlag; 1971.
[62] Gurtin ME, Fried E, Anand L. The mechanics and thermodynamics of continua. Cambridge University Press; 2010.
[63] Murphy KP. Machine learning: a probabilistic perspective. MIT Press; 2012.
[64] Marquardt DW. An algorithm for least-squares estimation of nonlinear parameters. J Soc Ind Appl Math 1963;11(2):431–41.
[65] Bottou L. Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010. Springer; 2010. p. 177–86.
[66] Gentle JE. Theory of statistics. George Mason University; 2013.
[67] Blei DM, Kucukelbir A, McAuliffe JD. Variational inference: A review for statisticians. J Am Stat Assoc 2017;112(518):859–77.
[68] Abadi M et al. Tensorflow: A system for large-scale machine learning. 12th USENIX symposium on operating systems design and implementation ({OSDI} 16), 2016.
[69] McKay MD, Beckman RJ, Conover WJ. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 2000;42(1):55–61.
[70] Aggarwal CC. Neural networks and deep learning. Springer; 2018. p. 978–83.