Computational Mechanics (2019) 64:281–305
https://doi.org/10.1007/s00466-019-01716-0

ORIGINAL PAPER

Clustering discretization methods for generation of material performance databases in machine learning and design optimization

Hengyang Li[1] · Orion L. Kafka[1] · Jiaying Gao[1] · Cheng Yu[1] · Yinghao Nie[2] · Lei Zhang[3] · Mahsa Tajdari[1] · Shan Tang[2] · Xu Guo[2] · Gang Li[2] · Shaoqiang Tang[3] · Gengdong Cheng[2] · Wing Kam Liu[1]

Received: 25 February 2019 / Accepted: 30 April 2019 / Published online: 22 May 2019

Abstract
Mechanical science and engineering can use machine learning. However, data sets have remained relatively scarce; fortunately, known governing equations can supplement these data. This paper summarizes and generalizes three reduced order methods: self-consistent clustering analysis, virtual clustering analysis, and FEM-clustering analysis. These approaches have two-stage structures: unsupervised learning facilitates model complexity reduction and mechanistic equations provide predictions. These predictions define databases appropriate for training neural networks. The feed forward neural network solves forward problems, e.g., replacing constitutive laws or homogenization routines. The convolutional neural network solves inverse problems or is a classifier, e.g., extracting boundary conditions or determining if damage occurs. We will explain how these networks are applied, then provide a practical exercise: topology optimization of a structure (a) with non-linear elastic material behavior and (b) under a microstructural damage constraint. This results in microstructure-sensitive designs with computational effort only slightly more than for a conventional linear elastic analysis.

Keywords Machine learning · Reduced order modeling · Materials database · Heterogeneous materials · Multiscale design optimization

1 Introduction

Computational methods in materials mechanics have evolved with the development of computation tools. A recent advance in computer sciences is the development of the so-called "Big Data" era, where a combined explosion in the number of sensors and datapoints along side computational resources and methods have enabled tracking and using large databases to develop understanding of the world, often replacing smaller more targeted studies that may produce less generalizable results or lack key insights. Taken in the context of computational mechanics, we can develop data-driven computational tools that rely on vast amounts of background data to facilitate, e.g. real-time multiscale simulations for fast multistage material system design, in-the-loop mechanics for controls (e.g. in manufacturing).

In order to develop such data-driven computational tools, two primary areas of study have emerged:

(1) Generation of materials system databases for materials mechanics, typically using data compression, to reduce computational complexity.
(2) Utilization of the database for real-time response prediction and multi-stage design.

The first area arises because data science relies heavily on the size and reliability of the database available. Unlike traditional applications in data science, such as image detection/recognition or automatic control, extremely large and well defined dataset are generally unavailable, or merely inaccessible, in computational mechanics. Whether databases are constructed from experiments or computational

Hengyang Li, Orion L. Kafka and Jiaying Gao have contributed equally to this work.

✉ Wing Kam Liu
w-liu@northwestern.edu

[1] Northwestern University, 2145 Sheridan Road, Tech B224, Evanston, IL 60208-3109, USA

[2] State Key Laboratory of Structural Analysis for Industrial Equipment, Department of Engineering Mechanics, Dalian University of Technology, Dalian, China

[3] HEDPS and LTCS, College of Engineering, Peking University, Beijing, China

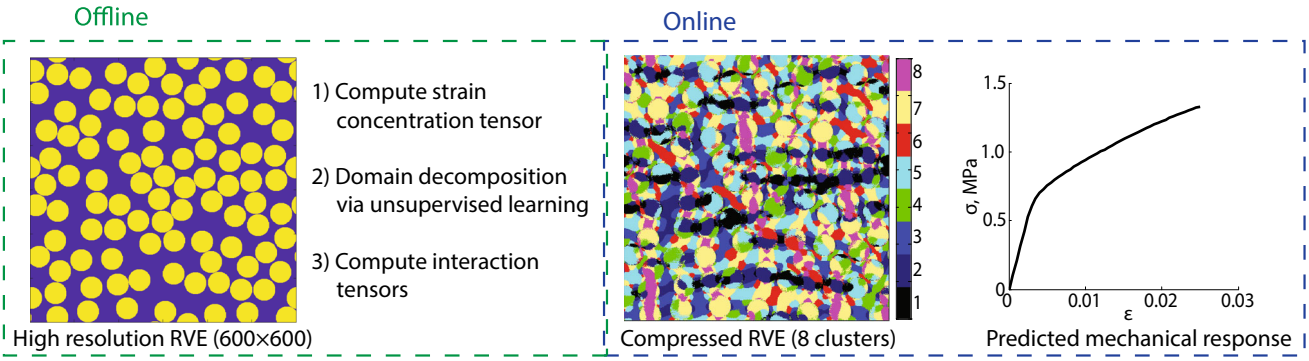

Fig. 1 Offline and online stages of two-stage clustering analysis methods. The offline stage contains three steps as shown in the figure, which will generate a compressed RVE database. The compressed RVE can be then used to predict mechanical responses of the RVE

models, the cost to generate data at a scale used in, e.g., image recognition has thus far been largely insurmountable. One approach to this challenge has been to use multiscale simulations of material systems using fast calculations of the overall stress of a representative volume element (RVE) for arbitrary far-field deformation loading. Many methods have been developed with the goal of finding an appropriate balance between cost and accuracy for such a problem; these are generally referred to as reduced order methods (ROMs), and many have been developed, e.g. [1–12].

The second area has often been addressed with methods from machine learning and neural networks to provide real-time prediction and multi-stage design. Data-driven methods have also been used to enhance computational mechanics by, for example, optimizing numerical quadrature [13] and replacing empirical constitutive laws with experiment data [14]. Recently, a deep material network method [15] was proposed which mimics neural networks topologically to link the micro material stiffness to the macro material stiffness. Once trained on a pre-simulated micro-macro stiffness database it can be used to compute, with significant speed-up, the overall stress of an RVE under arbitrary far-field deformation loading.

In this paper, three recent techniques in modeling microstructure based on data mining will be explored and generalized in Sect. 2. These kinds of fast methods address the first area: they can be used to generate the type of very large databases required for the pure or mechanics-enhanced machine learning outlined in the second portion of this paper. The second portion derives, in the language of mechanics, the workings of two different classes of neural networks, Sect. 3. One possible engineering application for these networks, topology optimization considering microstructure, is explored with detailed examples in Sect. 4. The some possible future directions for data-driven computational approaches are outlined to inspire further research in this emerging field within computational mechanics.

2 Overview of two-stage clustering analysis methods

Self-consistent Clustering Analysis (SCA) [9] and its close relatives Virtual Clustering Analysis (VCA) [16] and FEM Clustering Analysis (FCA) [17] are two-stage reduced order modeling approaches consisting of an offline data compression process and an online prediction process. This is concisely illustrated in Fig. 1 above. In the offline stage, the original high-fidelity Representative Volume Element (RVE) represented by voxels or elements is compressed into clusters. In the online stage, macroscopic loading is applied to the reduced order (clustered) RVE. The system of equations describing mechanical response is then solved on only the reduced representation as a boundary value problem. The notation used in this section is summarized in Table 1.

To define the boundary value problem, consider a material occupying $\Omega \subset \mathbb{R}^d$. The goal of homogenization is to find the macroscopic constitutive relation between a macroscopic stress

$$\boldsymbol{\sigma}^{\mathrm{M}} = \frac{1}{|\Omega|}\int_{\Omega} \boldsymbol{\sigma}(\boldsymbol{X})\mathrm{d}\boldsymbol{X}, \boldsymbol{X} \in \Omega \tag{1}$$

and a macroscopic strain

$$\boldsymbol{\varepsilon}^{\mathrm{M}} = \frac{1}{|\Omega|}\int_{\Omega} \boldsymbol{\varepsilon}(\boldsymbol{X})\mathrm{d}\boldsymbol{X}, \boldsymbol{X} \in \Omega, \tag{2}$$

where $|\Omega|$ is the total volume of the region Ω. We define mathematically the RVE problem as

$$\begin{cases} \nabla \cdot \boldsymbol{\sigma} = \boldsymbol{0}, \forall \boldsymbol{X} \in \Omega \\ \boldsymbol{\varepsilon} = \frac{1}{2}\left(\nabla \boldsymbol{u} + \boldsymbol{u}\nabla\right), \forall \boldsymbol{X} \in \Omega \\ \boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{\varepsilon}; \boldsymbol{X}), \forall \boldsymbol{X} \in \Omega \\ \text{Boundary conditions,} \end{cases} \tag{3}$$

where $\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{\varepsilon}; \boldsymbol{X})$ is a general microscale constitutive law. For the homogenization problem, boundary conditions have

Table 1 Notation used for two-stage clustering analysis methods

$\boldsymbol{X}$	Material point
$\boldsymbol{X}'$	Any other material point
$\boldsymbol{n}$	Normal to boundary
$\boldsymbol{\sigma}^{\mathrm{M}}$	Macroscale stress tensor
$\boldsymbol{\varepsilon}^{\mathrm{M}}$	Macroscale strain tensor
$\boldsymbol{\sigma}(\boldsymbol{X})$	Microscale stress tensor
$\boldsymbol{\varepsilon}(\boldsymbol{X})$	Microscale strain tensor
$\boldsymbol{s}$	Unit eigenstress
$\boldsymbol{e}$	Unit eigenstrain
$\tilde{\boldsymbol{\Gamma}}\left(\boldsymbol{X}, \boldsymbol{X}'\right)$	Green's operator
$\hat{\boldsymbol{\Gamma}}^0$	Fourier transform of the periodic Green's operator
I	Counting index for clusters
J	Another counting index for clusters
$\chi^I(\boldsymbol{X})$	Characteristic function for cluster I
$\boldsymbol{D}^{IJ}$	Interaction tensor for discretized Lippmann–Schwinger equation
$\boldsymbol{B}$	"Interaction tensor" for FCA
Ω	Material domain
Ω^I	Domain of the Ith cluster (subset of Ω)
c^I	Volume fraction of the Ith cluster
$\tilde{\boldsymbol{\varepsilon}}$	Reference material strain
$\tilde{\boldsymbol{C}}$	Reference material stiffness
$\tilde{\boldsymbol{S}}$	Compliance matrix for reference material
$\boldsymbol{A}$	Strain concentration tensor
$\boldsymbol{F}(\boldsymbol{X})$	Deformation gradient at point $\boldsymbol{X}$
$\boldsymbol{F^0}$	Reference deformation gradient
$\boldsymbol{A}'$	Deformation concentration tensor
λ^0, μ^0	Lamé's constants of the homogeneous stiffness tensor
$\boldsymbol{\xi}$	Fourier point
$\mathcal{F}, \mathcal{F}^{-1}$	Forward and inverse fast Fourier transform techniques
$\Delta\square$	Incremental form of an arbitrary variable $\square$
$\boldsymbol{r}^I$	Residual of the Ith cluster
$\boldsymbol{M}$	Jacobiam matrix of $\boldsymbol{r}$ with respect to $\Delta\boldsymbol{\varepsilon}$
$\boldsymbol{I}_4$	Fourth-order identity tensor
$\boldsymbol{C}^J_{alg}$	Tangent stiffness of the material in the Jth cluster
N_C	Number of clusters
N_F	Number of Fourier points
N_I	Number of integration points
N_E	Number of finite elements
E_{matrix}	Young's modulus of the matrix
ν_{matrix}	Poisson's ratio of the matrix
$E_{inclusion}$	Young's modulus of the inclusion
$\nu_{inclusion}$	Poisson's Ratio of the inclusion
$\sigma_{Y,\,matrix}$	Yield strength of the matrix
$\bar{\varepsilon}^{\mathrm{p}}$	Effective plastic strain (matrix)

to be chosen to satisfy the Hill–Mandel condition. In this paper, the periodic boundary conditions are used: $\boldsymbol{\varepsilon}$ periodic and $\boldsymbol{\sigma}\cdot\boldsymbol{n}$ anti-periodic on $\partial\Omega$.

2.1 Continuous and discretized Lippmann–Schwinger equation

By introducing a reference material with distributed elastic stiffness $\tilde{\boldsymbol{C}}(\boldsymbol{X})$, it has been shown that the RVE problem with periodic boundary conditions is equivalent to the integral equation [18]

$$\boldsymbol{\varepsilon}\left(\boldsymbol{X}\right) = \tilde{\boldsymbol{\varepsilon}}(\boldsymbol{X}) - \int_{\Omega} \tilde{\boldsymbol{\Gamma}}\left(\boldsymbol{X}, \boldsymbol{X}'\right) : \left(\boldsymbol{\sigma}\left(\boldsymbol{X}'\right) - \tilde{\boldsymbol{C}}(\boldsymbol{X}') : \boldsymbol{\varepsilon}\left(\boldsymbol{X}'\right)\right) \mathrm{d}\boldsymbol{X}' \tag{4}$$

where $\tilde{\boldsymbol{\varepsilon}}(\boldsymbol{X})$ is the strain in the reference material when applying the same loading and boundary conditions as the original RVE problem; $\boldsymbol{\sigma}\left(\boldsymbol{X}'\right) - \tilde{\boldsymbol{C}}(\boldsymbol{X}') : \boldsymbol{\varepsilon}\left(\boldsymbol{X}'\right)$ is the eigenstress applied to the reference material; $\tilde{\boldsymbol{\Gamma}}\left(\boldsymbol{X}, \boldsymbol{X}'\right)$ is the Green's operator associated with the reference material. The physical meaning of $-\tilde{\Gamma}_{ijkl}(\boldsymbol{X}, \boldsymbol{X'})$ is the strain component ε_{ij} at material point $\boldsymbol{X}$ if the unit eigenstress $\boldsymbol{s}^{kl}$ is applied at material point $\boldsymbol{X'}$, with the components of $\boldsymbol{s}^{kl}$ defined by $s^{kl}_{mn} = \delta_{km}\delta_{ln}$, where δ_{km} and δ_{ln} are Kronecker delta functions.

The integral equation given in Eq. (4) is known as the Lippmann–Schwinger equation, commonly seen to describe particle scattering in quantum mechanics. There is typically no explicit form of $\tilde{\boldsymbol{\Gamma}}\left(\boldsymbol{X}, \boldsymbol{X}'\right)$, unless the reference material is homogeneous.

The domain can be decomposed into several sub-regions, called clusters, distinguished mathematically by the characteristic function χ as shown in Eq. (5).

$$\chi^I(\boldsymbol{X}) = \begin{cases} 1, & \boldsymbol{X} \in \Omega^I \\ 0, & \text{otherwise} \end{cases}, \tag{5}$$

where $I = 1, 2, 3, \ldots, N_C$ denotes each cluster, and Ω^I is the subset of the volume within cluster I. These clusters are defined during the offline stage, as described in Sect. 2.2. This allows one to discretize the Lippmann–Schwinger equation, as:

$$\boldsymbol{\varepsilon}^I = \tilde{\boldsymbol{\varepsilon}}^I - \sum_{J=1}^{N_C} \boldsymbol{D}^{IJ} : \left(\boldsymbol{\sigma}^J - \tilde{\boldsymbol{C}}^J : \boldsymbol{\varepsilon}^J\right), \forall I \in \{1, \ldots, N_C\} \tag{6}$$

where $\square^I = \frac{1}{c^I|\Omega|}\int_\Omega \chi^I(\boldsymbol{X})\square(\boldsymbol{X})\mathrm{d}\boldsymbol{X}$ denotes the volume average of an arbitrary variable $\square$ in the Ith cluster; $c^I = \frac{|\Omega^I|}{|\Omega|}$ is the volume fraction of the Ith cluster where

the volume of the Ith cluster is given by $|\Omega^I|$. $\boldsymbol{D}^{IJ}$ is the interaction tensor given by

$$\boldsymbol{D}^{IJ} = \frac{1}{c^I\,|\Omega|}\int_\Omega\int_\Omega \chi^I(\boldsymbol{X})\chi^J\left(\boldsymbol{X}'\right)\tilde{\boldsymbol{\Gamma}}\left(\boldsymbol{X},\boldsymbol{X}'\right)\mathrm{d}\boldsymbol{X}\mathrm{d}\boldsymbol{X}'. \tag{7}$$

The physical meaning of $-\left(D^{IJ}_{ijkl}\right)$ is the average strain component ij in the Ith cluster if the uniform unit eigenstress component kl is applied in the Jth cluster.

Remark 1 If homogeneous reference material is used, as in SCA and VCA, $\tilde{\boldsymbol{\varepsilon}}^I = \boldsymbol{\varepsilon}^{\mathrm{M}}$.

Remark 2 A counterpart of Eq. (6), similar to the idea of FCA, can be expressed as

$$\boldsymbol{\sigma}^I = \tilde{\boldsymbol{\sigma}}^I - \sum_{J=1}^{N_C}\boldsymbol{B}^{IJ}:\left(\boldsymbol{\varepsilon}^J - \tilde{\boldsymbol{S}}^J:\boldsymbol{\sigma}^J\right), \tag{8}$$

where $\tilde{\boldsymbol{S}}$ is the compliance matrix of the reference material; $\tilde{\boldsymbol{\sigma}}$ is the stress in the reference material when applying the same loading and boundary conditions as the original RVE problem. $\boldsymbol{\varepsilon}^J - \tilde{\boldsymbol{S}}^J:\boldsymbol{\sigma}^J$ is the volume average eigenstrain in the Jth cluster. The physical meaning of $-B^{IJ}_{ijkl}$ is the average stress component ij in the Ith cluster if the uniform unit eigenstrain component kl is applied in the Jth cluster. Note that the reference material for FCA is the elastic state of the original RVE, instead of a homogeneous reference material as used by the other two methods here.

Remark 3 The relationship between $\boldsymbol{B}^{IJ}$ and $\boldsymbol{D}^{IJ}$ is given by

$$\tilde{\boldsymbol{S}}^I:\boldsymbol{B}^{IJ} = -\boldsymbol{D}^{IJ}:\tilde{\boldsymbol{C}}^J. \tag{9}$$

This can be proven by noting that the effect of applying any unit eigenstrain kl in some cluster J of the reference material is equivalent to that of applying eigenstress $-\tilde{\boldsymbol{C}}:\boldsymbol{e}^{kl}$ in the same cluster.

In the online stage, SCA solves the incremental, discretized Lippmann–Schwinger equation, Eq. (6), with arbitrary external loading conditions. These can either be of the fixed strain increment $\Delta\boldsymbol{\varepsilon}^{\mathrm{M}}$ type or be of the fixed stress increment $\Delta\boldsymbol{\sigma}^{\mathrm{M}}$ type.

2.2 Offline: clustering and interaction tensor

The offline stage consists of three primary steps: (1) data collection, (2) unsupervised learning (e.g. clustering), and (3) pre-computation of the interaction of clusters. Computation of the linear elastic response (data collection) and subsequent clustering based on that response are conducted identically for each of the three two-stage methods presented below: the same voxel mesh and clusters are used in the examples for all three methods. The difference between methods comes in the computation of the interaction tensor.

2.2.1 Data collection

Data collection provides information used to construct the reduced representation of the system. It typically involves some computation of the response of a fully resolved system, perhaps with a simplified material model, over a limited set of loading cases. One measure of mechanical response that could be collected is the strain concentration tensor used in micromechanics $\boldsymbol{A}$, defined by $\boldsymbol{\varepsilon}(\boldsymbol{X}) = \boldsymbol{A}(\boldsymbol{X}):\boldsymbol{\varepsilon}^{\mathrm{M}}(\boldsymbol{X})$, which maps between the far-field or applied strain, $\boldsymbol{\varepsilon}^{\mathrm{M}}$ and the strain measured at point in the domain, $\boldsymbol{\varepsilon}$. See [9] for more details.

The choice of the strain concentration tensor as given above is often suitable, but it depends on the relevant details of the problem. For example, at finite deformations one might consider using the deformation concentration tensor: $\boldsymbol{A}'(\boldsymbol{X}) = \frac{\partial \boldsymbol{F}(\boldsymbol{X})}{\partial \boldsymbol{F}^0}$ where $\boldsymbol{F}(\boldsymbol{X})$ is the deformation gradient at any given point $\boldsymbol{X}$ within the domain and $\boldsymbol{F}^0$ is the macroscopic deformation corresponding to the boundary conditions. See [12] and [10] for more detailed descriptions of the finite strain formulation. Alternatively, if the elastic and plastic material responses differ substantially, e.g. if one is isotropic and the other anisotropic, including information about the plastic part of the deformation might be desirable.

2.2.2 Clustering

The goal of clustering is to reduce the number of degrees of freedom required to represent the system while minimizing the loss of information about the mechanical response. One way of doing this, as alluded to above, is by grouping material points within the domain of interest. If one can assume that the material response within each group is identical, the evolution of the domain can be determined by solving for the response of each group rather than each material point.

Using the data generated during the collection phase, one of the many clustering (or unsupervised learning) techniques might be applied to optimized the domain decomposition. In the following examples, Self-Organizing Maps (SOMs) are employed following the method outlined by [19]. This is illustrated in Fig. 1, where eight clusters (four in each phase) are constructed. The clustering process assigns each material point with a cluster ID, such that clusters are labelled $1, 2, \ldots, N_C$.

The k-means clustering method has also been used, e.g. in [9–11]. More elaborate clustering schemes might also be

considered. Ongoing efforts include "adaptive clustering" schemes that mimic adaptive FE methods in their ability to evolve as deformation progresses, and "enriched" machine learning, whereby *a priori* information from mechanics about the deformation fields outside the bounds of the data collected in Step 1 are used to guide or bound the unsupervised learning. Another unexplored future direction might use a "feature-based" machine learning that includes microstructure information in addition to mechanics information.

2.2.3 Interaction tensor

The interaction tensor describes the impact each cluster has on each of the other clusters. Once the clustering process is completed, the interaction tensor can be explicitly computed. Importantly, the integral part only has to be computed once during the offline stage. Only the results of that calculation are then used for the online stage. Three ways to compute the interaction tensor, one used by each of the methods highlighted here (though these are not exclusive to each), are:

SCA: Fourier transform for D^{IJ} With periodic boundary conditions and a homogeneous reference material, the Green's operator has a simple expression in Fourier space, given by

$$\hat{\Gamma}^0_{ijkl}(\boldsymbol{\xi}) = \frac{\delta_{ik}\xi_j\xi_l}{2\mu^0\,|\boldsymbol{\xi}|^2} - \frac{\lambda^0}{2\mu^0\left(\lambda^0+2\mu^0\right)}\frac{\xi_i\xi_j\xi_k\xi_l}{|\boldsymbol{\xi}|^4} \tag{10}$$

where $\hat{\Gamma}^0_{ijkl} = \mathcal{F}(\boldsymbol{\Gamma}^0)$ is the Fourier transform of a periodic Green's operator $\boldsymbol{\Gamma}^0$; λ^0 and μ^0 are the Lamé's constants of the homogeneous stiffness tensor; $\boldsymbol{\xi}$ is the Fourier point. Then the interaction tensor can be calculated with

$$\boldsymbol{D}^{IJ} = \frac{1}{c^I\,|\Omega|}\int_\Omega \chi^I(\boldsymbol{X})\mathcal{F}^{-1}\left(\mathcal{F}(\chi^J)\mathcal{F}(\boldsymbol{\Gamma}^0)\right)\mathrm{d}\boldsymbol{X}, \quad \forall I, J \in \{1,\ldots,N_C\} \tag{11}$$

using the fast Fourier transform (FFT) technique. The computational complexity is $O((N_C)^2(N_F)\log(N_F))$, where N_F is the number of Fourier points used in the FFT calculation.

VCA: numerical integration for D^{IJ} With an infinite homogeneous reference material, the Green's operator can be expressed in real space; however, the expression is lengthy so has not been reproduced here. The interested reader will find it in [16] and [20]. Numerical integration is the most straightforward method to compute the integral equation given in Eq. (7). The computational complexity is $O((N_I)^2)$, where N_I is the number of integration points used. In [20], a fast method is proposed to approximate D^{IJ}.

FCA: finite element method for B^{IJ} Based on the physical interpretation of the interaction tensor, the finite element method can also be used. By applying uniform unit eigenstrain component kl in the Jth cluster, the average stress can be computed for all clusters, resulting in B^{IJ}_{ijkl} for all $I = 1,\ldots,N_C$. Thus, the computational complexity is $O(6(N_C)(N_E))$, where N_E is the number of finite elements used. The tensor $\boldsymbol{B}^{IJ}$ is similar to the interaction tensor $\boldsymbol{D}^{IJ}$, although $\boldsymbol{B}^{IJ}$ is determined by applying strains rather than stresses.

2.3 Online: reduced order response prediction

Once the interaction tensor database is prepared, the discretized Lippmann–Schwinger equation defined in Eq. (6) is solved in the online stage. An incremental form of Eq. (6) is given by

$$\Delta\boldsymbol{\varepsilon}^I = \Delta\tilde{\boldsymbol{\varepsilon}}^I - \sum_{J=1}^{N_C}\boldsymbol{D}^{IJ} : \left(\Delta\boldsymbol{\sigma}^J - \tilde{\boldsymbol{C}}^J : \Delta\boldsymbol{\varepsilon}^J\right), \quad \forall I \in \{1,\ldots,N_C\}, \tag{12}$$

where $\Delta\tilde{\boldsymbol{\varepsilon}}^I$ is the applied incremental reference strain. The incremental stress $\Delta\boldsymbol{\sigma}^I$ is a function of the incremental strain $\Delta\boldsymbol{\varepsilon}^I$ according to the local material constitutive laws. So the unknowns of Eq. (12) are the strains in each cluster $\{\Delta\boldsymbol{\varepsilon}\} = \{\Delta\boldsymbol{\varepsilon}^1,\ldots,\Delta\boldsymbol{\varepsilon}^{N_C}\}$. For nonlinear microscale constitutive laws, Eq. (12) is nonlinear and has to be solved iteratively. Newton's iterative method is used by SCA and VCA, while a different iterative method is used by FCA. The residual form of Eq. (12) is given by

$$\boldsymbol{r}^I = \Delta\boldsymbol{\varepsilon}^I - \Delta\tilde{\boldsymbol{\varepsilon}}^I + \sum_{J=1}^{N_C}\boldsymbol{D}^{IJ} : \left(\Delta\boldsymbol{\sigma}^J - \tilde{\boldsymbol{C}}^J : \Delta\boldsymbol{\varepsilon}^J\right), \quad \forall I \in \{1,\ldots,N_C\} \tag{13}$$

then the Jacobian matrix $\{\boldsymbol{M}\} = \frac{\partial\{\boldsymbol{r}\}}{\partial\{\Delta\boldsymbol{\varepsilon}\}}$ for the Newton's method is given by

$$\boldsymbol{M}^{IJ} = \frac{\partial\boldsymbol{r}^I}{\partial\Delta\boldsymbol{\varepsilon}^J} = \delta_{IJ}\boldsymbol{I}_4 + \boldsymbol{D}^{IJ} : (\boldsymbol{C}^J_{\mathrm{alg}} - \tilde{\boldsymbol{C}}), \quad \forall I, J \in \{1,\ldots,N_C\} \tag{14}$$

where $\boldsymbol{I}_4$ is the fourth-order identity tensor. The tangent stiffness of the material in the Jth cluster is $\boldsymbol{C}^J_{\mathrm{alg}}$. An alternative way to solve for the local mechanical responses, proposed by [21], is to minimize the complementary energy of the clustering-based system.

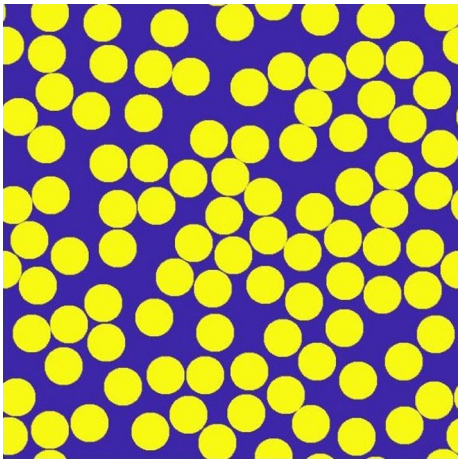

Fig. 2 Geometry for example problem: a two-phase periodic composite represented in 2D with plane strain where blue is a continuous matrix phase and yellow represents inclusions. (Color figure online)

2.4 Comparison of the methods

In order to compare the accuracy and efficiency of each method, a 2D plane strain model for a two-phase material is constructed and shown in Fig. 2. The 2D mesh contains 600×600 square pixels. The inclusion area fraction is 51%. Material constants of the matrix and the inclusion are given in Table 2. The yeild surface is the von Mises criterion as shown in Eq. (15). The hardening law of the matrix material is given in Eq. (16). Note that this will be approximated by a non-linear elastic behavior for monotonic loading in future sections.

$$f = \bar{\sigma} - \sigma_{Y,matrix}\left(\bar{\varepsilon}^{p}\right) \leq 0 \tag{15}$$

$$\sigma_{Y,matrix} = \begin{cases} 0.50 + 5\bar{\varepsilon}^{\mathrm{p}}, & 0 < \bar{\varepsilon}^{\mathrm{p}} \leq 0.04 \\ 0.62 + 2\bar{\varepsilon}^{\mathrm{p}}, & 0.04 < \bar{\varepsilon}^{\mathrm{p}} \end{cases} \tag{16}$$

The equivalent von Mises stress is $\bar{\sigma}$. The yield stress $\sigma_{Y,matrix}$ is given by the hardening law in Eq. (16) with equivalent plastic strain $\bar{\varepsilon}^{\mathrm{p}}$. Strain from 0 to 0.05 is prescribed in the x-direction and zero strain is enforced in the y- and xy-directions. A direct numerical simulation (DNS) of the microstructure under these loading conditions is performed using the FEM and the effective von Mises stress is recorded for later comparison to the reduced order model results.

A one-time data-compression of the microstructure is performed following the steps outlined in Sect. 2.2, using the strain concentration tensor. The resulting clustering of the microstructure is shown in Fig. 3. The interaction tensor for each method is computed using the aforementioned algorithms in Sect. 2.2.2.

The different methods, used to compute the interaction tensors, each result in a slightly different form of the tensor. There are strong similarities—after all, the same microstructure and clustering is used—though the details differ. Figure 4 shows a magnitude plot of each of the three methods, where the magnitude represents the effect of each stress component in the Jth cluster on the corresponding strain component in the Ith cluster. The trends of the magnitudes shown in Figs. 4 and 5 suggest that inter-cluster interaction is not as strong for VCA as for SCA and FCA, although this difference is relatively minor. The strongest interactions are intra-cluster, as shown by the peaks along the diagonal. FCA has two distinct regions of peaks, corresponding to the set of clusters in the inclusion and in the matrix. The tensor is constructed in an ordered way, which results in these two distinct sets of clusters. This is unlikely to change the overall solution accuracy.

Once the microstructural database is created and the interaction tensor has been computed, the online prediction is

Table 2 Material constants for matrix and inclusion

E_{matrix} (MPa)	ν_{matrix}	$E_{\mathrm{inclusion}}$ (MPa)	$\nu_{\mathrm{inclusion}}$	Area fraction of inclusion
100.0	0.30	500.0	0.19	0.51

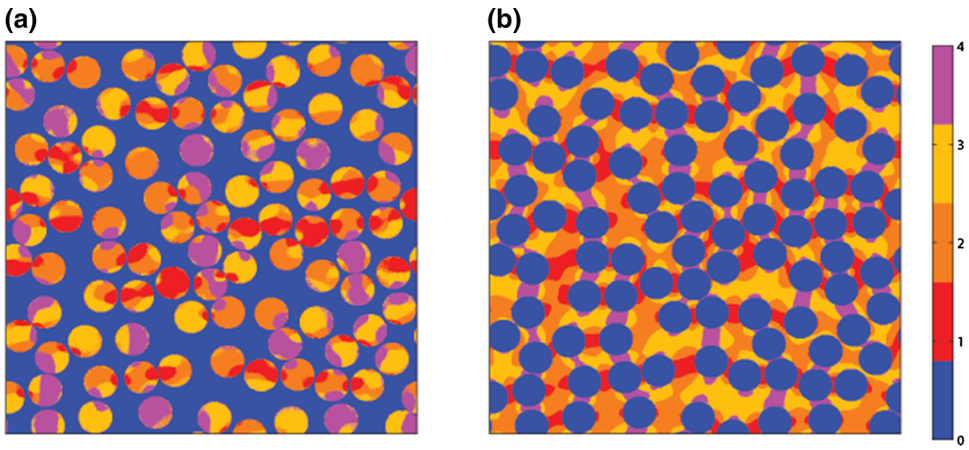

Fig. 3 Color contours showing clusters distribution in **a** the inclusions and **b** the matrix. Note that clusters need not be spatially connected. (Color figure online)

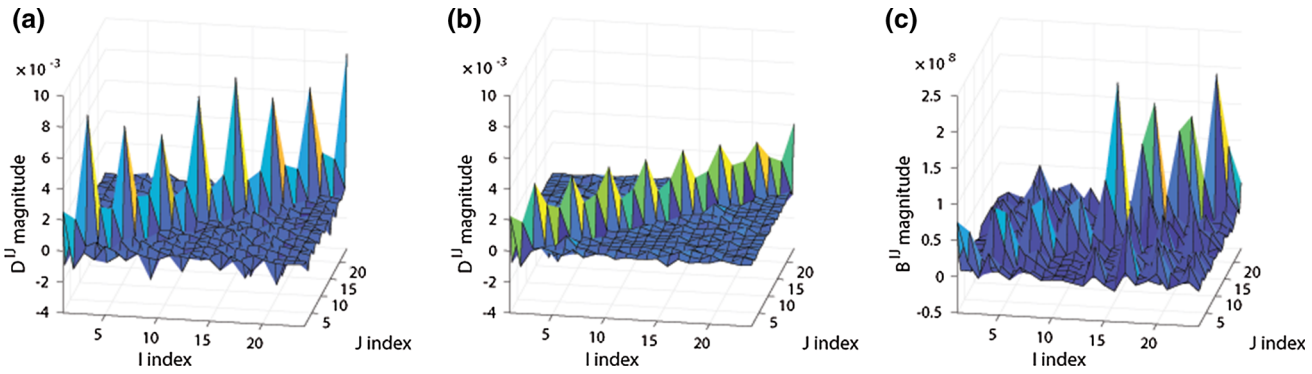

Fig. 4 Component-wise magnitude plots for **a** $\boldsymbol{D}^{IJ}_{\mathrm{SCA}}$, **b** $\boldsymbol{D}^{IJ}_{\mathrm{VCA}}$, **c** $\boldsymbol{B}^{IJ}_{\mathrm{FCA}}$. Spikes along the diagonal direction for all three interaction tensor surface plots suggest self interaction has more contribution than the rest of clusters in cluster-wise stress increment. $\boldsymbol{D}^{IJ}_{\mathrm{SCA}}$ and $\boldsymbol{D}^{IJ}_{\mathrm{VCA}}$ have the similar magnitude along their diagonal direction due to the homogeneous reference material assumption. $\boldsymbol{B}^{IJ}_{\mathrm{FCA}}$ has different magnitudes for matrix and inclusion phase along the diagonal direction, implicitly representing a heterogeneous reference material

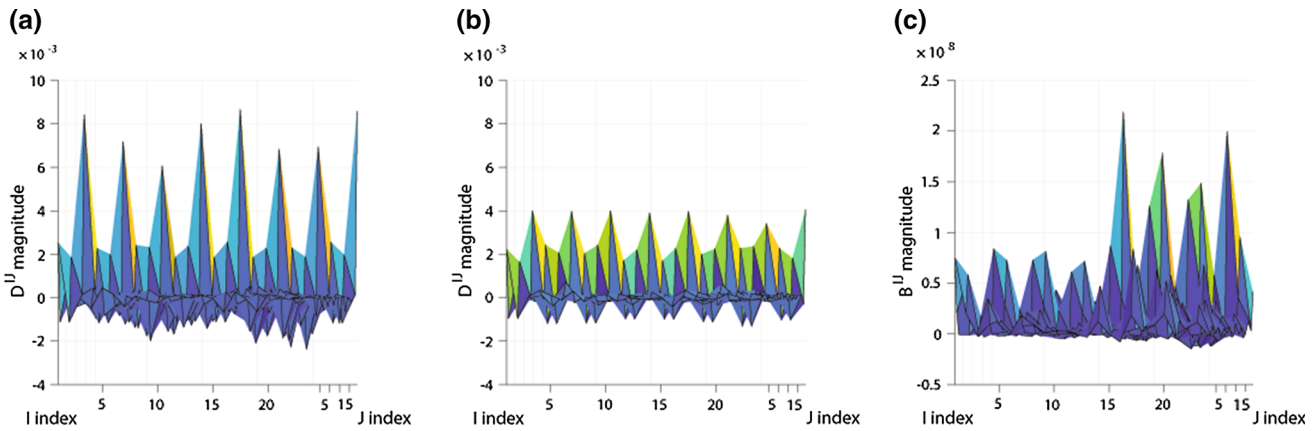

Fig. 5 Plots for **a** $\boldsymbol{D}^{IJ}_{\mathrm{SCA}}$, **b** $\boldsymbol{D}^{IJ}_{\mathrm{VCA}}$, **c** $\boldsymbol{B}^{IJ}_{\mathrm{FCA}}$ in profile; note that for FCA the two regions correspond to different physical domains (matrix and inclusion)

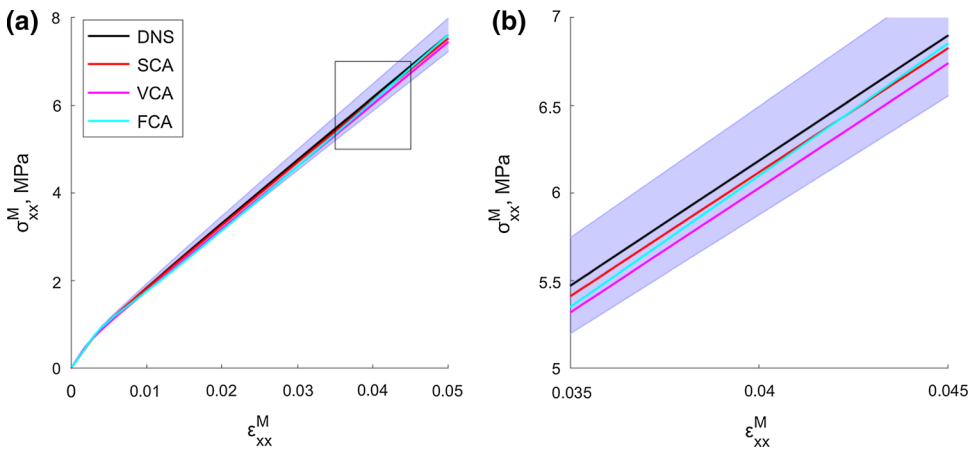

Fig. 6 Plots of σ^{M}_{xx} vs. $\varepsilon^{\mathrm{M}}_{xx}$. All three methods performed well, with predictions laying within 5% of the reference solution[1]

performed. Figure 6 shows that the ROM results for stress in the x-direction in all three cases are within 5% of the DNS results,[1] with small differences between the three ROMs. Fig. 6b highlights the slight differences between the three methods: SCA follows the same trend as the DNS, but is slightly softer; the response predicted by VCA is softer still, as a result of the constant reference material assumption; the trend exhibited by FCA is slightly different from the DNS

[1] A 5% deviation from the reference solution (DNS) is indicated by the blue shaded region in Figs. 6 and 7.

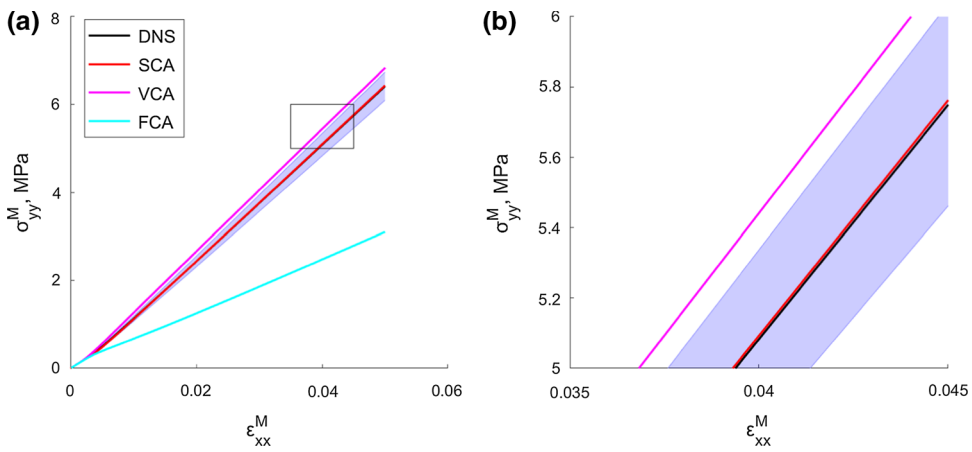

Fig. 7 Plots of σ^{M}_{yy} vs. $\varepsilon^{\mathrm{M}}_{xx}$. SCA has the best agreement with the DNS solution in this case, whereas VCA and FCA deviate from the DNS solution. The causes for such deviation and improvements in VCA and FCA will be explored in the future

overall, although still quite close in value. Figure 7, σ^{M}_{yy} plotted against $\varepsilon^{\mathrm{M}}_{xx}$, shows that SCA has the best agreement with the DNS solution, and is the only prediction within 5% of the reference solution[1]. Note that VCA has different boundary conditions than the DNS solution, i.e. it uses a fictitious surrounding domain. Some deviation from a DNS result with periodic boundary conditions is therefore expected.

3 Machine learning on databases generated with predictive ROMs

Neural networks are a specific class of machine learning algorithms, which in the most basic form appear similar to regression analysis. In practice, these methods involve modifying input data through a series of functions to obtain output data. The exact series of functions and their weights and forms depend on the application. These methods can provide an increase in speed over more conventional approaches, once the algorithm has been appropriately trained. In order to have a highly effective modeling approach based on machine learning, rich databases of mechanical response information are required to perform that training. Developing such a database with experiments is intractable, particularly for design of new materials or material systems where no material performance information exists. The fast, predictive models (SCA, VCA, FCA) outlined in Sect. 2 are thus desirable for quickly populating relatively large materials databases, as will be shown. This enables the use of machine learning algorithms in multiscale design, where simultaneously application and satisfaction of criteria and constraints governing the material microstructure and component-level macrostructure is required.

Feed forward neural networks (FFNNs) [13,22,23] were the first neural networks developed [24]. The FFNN was designed to learn complex input-output relations. As such, FFNNs can be used to replace conventional constitutive laws; this is particularly appealing when descriptions of the homogenized behavior of a material is complex and/or difficult to obtain. The basic structure of an FFNN consists of an input layer, hidden layers, and an output layer. Every pair of neurons in neighboring layers have a weighted connection. Each neuron in hidden layers and the output layer has a bias. In FFNNs, neurons in the same layer are not connected. In the learning procedure, the connection weights are changed following a predefined set of rules, such as with back propagation. Funahashi [25] and Hornik et al. [26] proved that three hidden layers in an FFNN is sufficient to learn any non-linear continuous function.

Early efforts to apply machine learning to mechanics used a computation and knowledge representation paradigm, which is actually a type of feed forward neural network, to directly "learn" material behavior by training from analytic and experimental data. One early work in 1991, for example, was Ghaboussi et al. [27] which applied a back-propagation neural network to model the behavior of concrete in plane stress under monotonic biaxial loading and compressive uniaxial cyclic loading. Furukawa et al. [28] used an implicit constitutive model and proposed an implicit viscoplastic constitutive model using neural networks. They generalized inelastic material behaviors in a state-space representation and constructed the state-space form by a neural network using input-output data sets. Similar approaches can also be found in other relatively early works, e.g. [29–33]. Once an RVE model is established, for example, an FFNN can be trained on that data, and a concurrent multiscale scheme to directly connect microstructure to the macro-scale material response might be achieved, with the FFNN replacing the RVE or constitutive law to describe the response at each material point [34–36].

Neural networks have been studied with the goal of integration with multiscale methods. In such cases, some parts of numerical simulations are replaced with neural networks to better utilize their merits. For example, Ghaboussi et al.

[37] introduced a self-learning finite element procedure. The method was further developed by Shin et al. [38,39], and Gawin et al. [40]. Lefik and Schrefler [41,42] introduced an artificial neural network as an incremental non-linear constitutive model for a finite element code. Recently, B.A. Le et al. [43] proposed a decoupled computational homogenization method for nonlinear elastic materials. Satyaki et al. [44] introduced a new manifold-based reduced order model for nonlinear problems. Cecen et al. [45] use a 3D Convolutional neural network to link material structure and properties. Wang and Sun [46] proposed a new meta-modeling framework based on reinforcement learning. With the development of numerical methods and the increasing interest in multiscale simulation, integration of multiscale simulation and neural networks is continuously developing.

3.1 Motivation

In design optimization, one might expect there to be many calls to the material subroutine (e.g. one for each element for each load for each design iteration). Thus running ROMs for RVEs might still be time consuming, compared to a model predicting RVE stress responses given a strain state within milliseconds, or a model providing strain state given microstructure stress contours within milliseconds. To tackle this issue, we propose replacing the ROMs by neural networks trained on the RVE responses computed with SCA for micro-stress and macro-stress. These networks preserve microstructure information and are engineered to:

- Predict macro (homogenized) stress or micro (local) stress given a macro-strain using an FFNN for one RVE. In this case, the FFNN plays the role of traditional material constitutive equations and homogenization (to compute the macro-stress). We will denote these $\mathscr{F}^{micro}_{FFNN}$ and $\mathscr{F}_{FFNN}$ and explain them in Sect. 3.4.
- Predict macro-strain given any micro-stress distributions using a convolutional neural network (CNN). This is the inverse of the FFNN. The input is a stress distribution within an RVE domain. The output is the macro strain loading applied to this RVE (the boundary conditions). We will denote this $\mathscr{F}_{CNN}$ and it will be explained in Sect. 3.6.
- Compute RVE damage by comparing local von Mises stress to a stress threshold. A CNN is trained to identify the onset of damage within an RVE. In this case, the CNN acts as a classifier, which identifies whether or not the applied macro-strain will cause microstructural damage. We will denote this $\mathscr{F}^{classify}_{CNN}$ and it will be introduced in Sect. 3.7.

3.2 Database generation for machine learning using SCA

The ROMs presented in this paper can be used to generate a database for machine learning; in this case, we are using SCA with 8 clusters, as in Sect. 2, for demonstration. In general, such databases contain N_T training samples and N_V validation samples. For the RVE model shown in Sect. 2, a database with $N_T = 1000$ strain–stress pairs is computed for a nonlinear elastic material by randomly sampling 200 terminal states with four sub-loading steps each, as shown in Fig. 8. Similarly, $N_V = 150$ strain–stress pairs (30 final states, with four intermediate steps) are generated for validation. A monotonically increasing load is assumed and all the final points are confined to a spherical space with a radius of 0.05. Table 3 shows the reduction in time required to build a database achieved by using SCA, versus FEM or FFT for this example case. Without a reduced order model, the generation of microstructure database takes days using FEM or FFT. The database contains $N_T = 1000$ pairs of macroscopic strains and stresses $\boldsymbol{\varepsilon}^{\mathrm{M},s}$, $\boldsymbol{\sigma}^{\mathrm{M},s}$, and local (micro) stress $\boldsymbol{\sigma}^s(\boldsymbol{X})$, where $s = 1, 2, \ldots, N_T$. In this case, we only consider a plane strain problem and

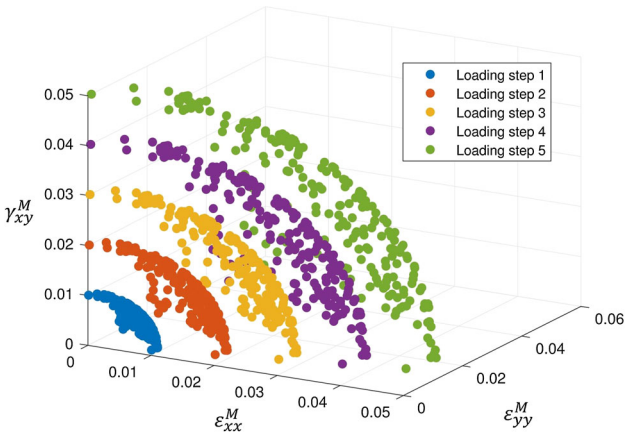

Fig. 8 Random samples of strain state; two hundred final states were selected, and four evenly spaced intermediate steps to reach the final states were recorded for a total of $N_T = 1000$ samples. All strain states will be applied to the RVE to generate corresponding stress states

Table 3 Comparison of time required to generate a microstructure database for nonlinear elastic RVE responses with $N_T = 1000$ samples of strain states

Method	Total time (s)	Speedup over FEM
FEM	2.04×10^7	–
FFT	3.01×10^5	68
SCA (4 + 4 clusters)	Offline: 13.0 + online: 2.16×10^3	9400

SCA requires only one single workstation to generate the database

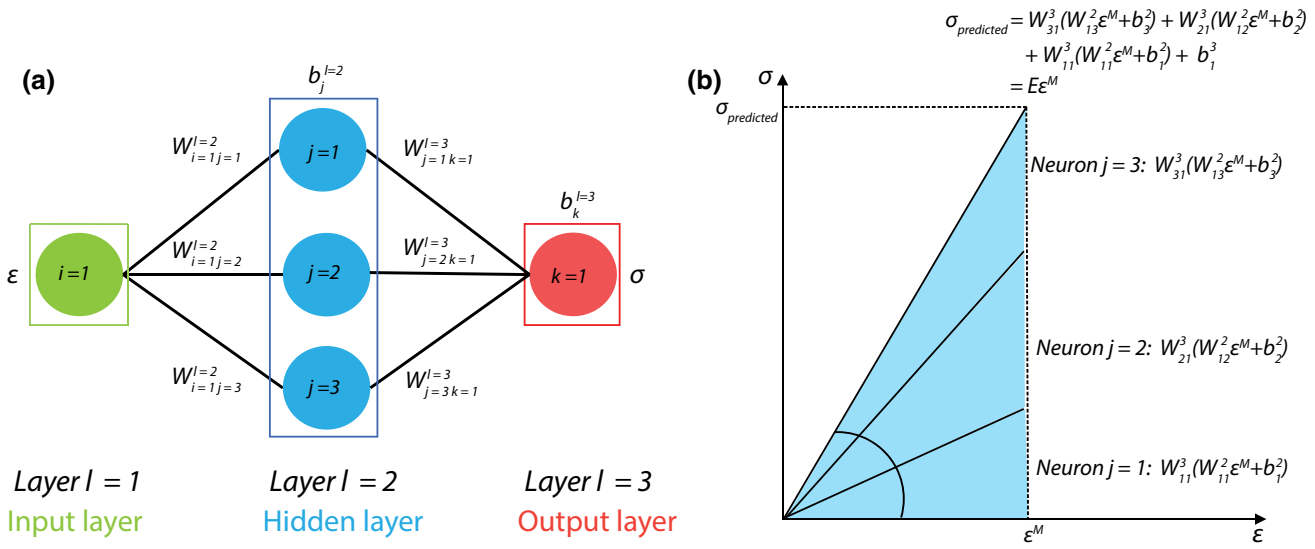

Fig. 9 **a** Illustration of an FFNN network with one hidden layer for a linear elastic example; the collective function of the weights and biases connecting the input layer (green), the hidden layer (blue), and the output layer (red) is that of Young's modulus E. **b** A stress-strain diagram, showing how the input strain is interpreted by the FFNN for a linear elastic case using linear activation functions and zero biases. (Color figure online)

do not consider stress in the z direction. SCA reduces the total computational time by two orders of magnitude compared to FFT and by four orders of magnitude compared to FEM.

3.3 Feed forward neural networks

In order to illustrate the structure of feed forward neural networks (FFNNs), a simplified one dimensional example is presented. In linear elasticity, stress is related to strain by the material stiffness; this can be generically defined as a mapping. The overall structure of a neural network can also be described as a mapping, i.e.:

$$\begin{cases} Constitutive\ equation:\ stress = \mathscr{F}_{Constitutive}(strain) \\ Neural\ network\ mapping:\ stress = \mathscr{F}_{FFNN}(strain) \end{cases} \tag{17}$$

where $\mathscr{F}_{FFNN}$ is the FFNN that uses strain state ε as input, and generates stress state σ as the output. The structure of a simple FFNN is shown in Fig. 9a. The notation used in this figure and throughout this section is defined in Table 4. For this illustration case, only one sample is considered, hence $s = 1$ and all variables are written without the superscript s. A general FFNN contains neurons (the circles) and weights (black lines). In general, an FFNN has one input layer, one output layer, and multiple hidden layers. Each layer may have multiple neurons; for the input and output layers, these are simply the input and output values. In the simplest case, an FFNN would have one input neuron, one hidden neuron, and one output neuron. For 1D linear elastic stress analysis in such a case, the input neuron would be strain, the hidden neuron would act as a multiplicative, functional decomposition of the stiffness that recovers the total stiffness required to map strains to stress in the output neuron. Generalizing this slightly, we might consider an FFNN with three hidden neurons, as shown in Fig. 9. Each neuron has only one value. The first neuron is simply the strain:

$$a^{l=1}_{i=1} = \varepsilon \text{ (input layer)} \tag{18}$$

Table 4 Notation table of variables used in the feed forward neural network

$\varepsilon^{M,s}_j$	Macroscale strain tensor components, $s = 1, \ldots, N_T$
s	Counting index for number of samples (training or validation, depending on context)
l	Counting index for number of layers
i	Counting index for neurons in a given layer
j	Counting index for neurons in another layer
N_T	Number of training samples
N_L	Number of layers in the neural network
$N_N(l)$	Number of neurons in layer l
W^l_{ij}	Weight connecting the ith neuron in layer $l-1$ to the jth in layer l
b^l_j	Bias of the jth neuron in layer l
$a^{l,s}_j$	Neuron value for jth neuron in lth layer and for sth sample
$\mathscr{A}$	Activation function
$\mathscr{F}_{FFNN}$	Feedforward neural network function

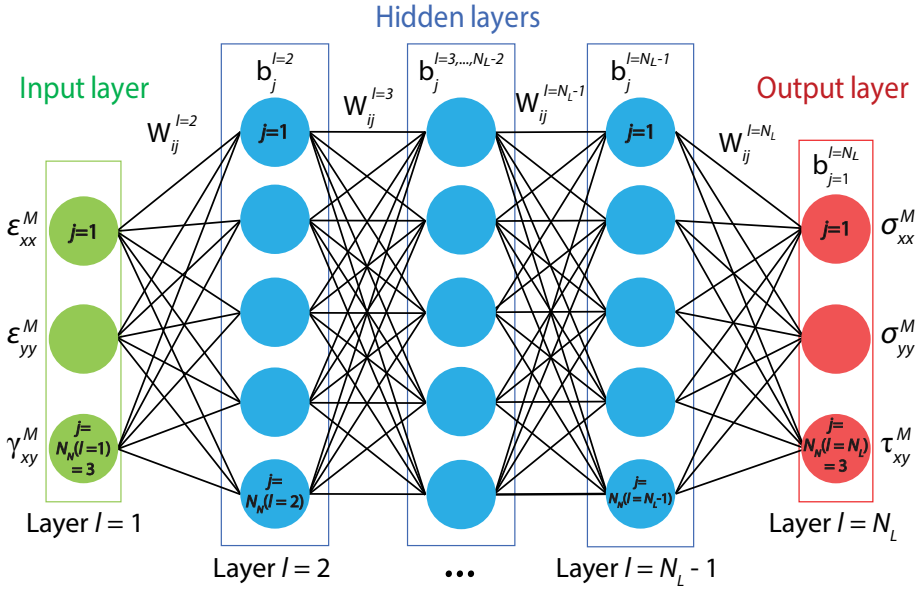

Fig. 10 Illustration of an FFNN with multiple hidden layers; N_L: index of layers, $N_N(l)$: number of neurons in layer l. The formulation of the FFNN is given in Eq. (21) with associated interpretation of the FFNN structure. The indices i and j representing neuron id in previous layer and current layer, e.g., $W_{12}^{l=2}$ is the weight between neuron 1 in layer $l = 1$ and neuron 2 in layer $l = 2$

The three neurons in the hidden layer take in this value, and each take on the value given by:

$$a_{j=1,2,3}^{l=2} = \mathscr{A}\left(\sum_{i=1}^{1} W_{ij}^{l=2} a_i^{l=1} + b_j^{l=2}\right) \text{ (hidden layer)} \quad (19)$$

where $\mathscr{A}$ is an activation function. In the training part, this paper uses the Sigmoid function [47]: $f(x) = \frac{1}{1+e^{-x}}$, and each neuron is computed using a different weight $W_{ij}^{l=2}$ and bias $b_j^{l=2}$, where i is the neuron in the previous layer (in this case, the input later) and j is the neuron in the current layer (in this case, the hidden layer). Finally, the overall response – the stress – is given by:

$$\begin{aligned}\sigma_{predicted} &= a_{k=1}^{l=3} \\ &= \sum_{j=1}^{3} W_{jk}^{l=3} a_j^{l=2} + b_k^{l=3} \text{ (output layer)} \end{aligned} \quad (20)$$

The combination of all the W and b terms, as well as the activation function, work as the fitting factors in a regression analysis. The activation function $\mathscr{A}$ is fixed for all neurons, and is used to condition the weighting factors. For the constitutive model outlined above, the overall results of the weights, bias and activation function would perfectly match the elastic modulus. If we only consider weights and the activation function a just a linear mapping, the function of each neuron in hidden layer is given explicitly in Fig. 9b where the bias is taken as zero for all neurons.

The physical interpretation of individual neurons is more complicated for non-linear responses, although the overall idea is the same. In this formulation, strain path dependence (i.e. plasticity) is impossible to capture, and the net result is an response map where one might think of the collection of neurons (weights, biases and activation functions) as an "instantaneous elastic modulus," or the slope of a line that relates strain to stress at the current point in strain.

Another example shown in Fig. 10 extends the previous example to consider a two-dimensional stress analysis, with many neurons per layer and several layers. The inputs are the three macroscale strain components $\varepsilon_{xx}^{\mathrm{M}}$, $\varepsilon_{yy}^{\mathrm{M}}$, γ_{xy}^{M}, in a plane strain problem. The outputs are the three macroscale stress components σ_{xx}^{M}, σ_{yy}^{M}, τ_{xy}^{M}. The stress component σ_{zz}^{M} is not considered. The samples (stress-strain pairs) from the database outlined above are used to train the neural network. After training, the FFNN can predict stresses when given strain inputs. In this example, MATLAB is used to build the FFNN and to train the neural network parameters [48].

In each layer of a general FFNN, each neuron takes the output value from each neuron in the preceding layer as inputs and gives a single output. This is repeated for each layer. Generalizing Eqs. (18), (19), and (20) to an arbitrary number of layers and neurons per layer results in Eq. (21), where the value of the jth neuron in layer l for the sth sample (either a training sample or prediction) can be expressed as:

$$a_j^{l,s} = \begin{cases} \varepsilon_j^{\mathrm{M},s}, \text{ if } l = 1 \text{ (input layer)} \\ \mathscr{A}\left(\sum_{i=1}^{N_N(l-1)} W_{ij}^{l} a_i^{l-1,s} + b_j^{l}\right), \\ \quad \text{if } l \in \{2, \ldots, N_L - 1\} \text{ (hidden layers)} \\ \sum_{i=1}^{N_N(l-1)} W_{ij}^{l} a_i^{l-1,s} + b_j^{l}, \text{ if } l = N_L \text{ (output layer)} \end{cases} \quad (21)$$

where the final layer gives the estimated stress:

$$\sigma_{predicted,j}^{\mathrm{M},s} = a_j^{N_L,s}. \quad (22)$$

The outputs $a_j^{l,s}$ of each layer possesses similar physical meaning as explained for the 1D case. The input strain components are represented by $l = 1$,$a_j^{l,s}$ for the sth sample, and $l = 2, \ldots, N_L - 1$, $a_j^{l,s}$ represents an estimate of the the nonlinear stress responses of the microstructure. The activation functions and weights of layer $l = 2, \ldots, N_L - 1$ play a roughly similar role to the classical definition of the tangent modulus in solid mechanics. The hidden layers take in strain components and produce an estimate of the non-linear stress responses. During the training process, the non-linear relationship between stress and strain is gradually "learned" by those hidden layers. In the last layer, $l = N_L$, $a_j^{l,s}$ represents the predicted stress components. The predicted stress components are produced through the regression operation in the output layer, as shown in Eq. (21). The weights and bias of the output layer correct the prediction generated from hidden layers, and produce accurate nonlinear stress responses. In order to make the concept clear, one might consider the hidden layer as unitless values operating on intermediate strain values, while the units of W and b in output layer are those of stress (e.g. MPa in the example problem). The FFNN can learn nonlinear elastic material behaviors due to following two key factors: (1) hidden layers approximate the material nonlinear elastic responses as a traditional constitutive model would do, as described in Eq. (17); (2) the output layer corrects the predicted nonlinear responses for improved prediction accuracy.

3.4 Feed forward neural network with database generated by SCA

In this case study, an FFNN is trained with data generated using SCA. Using the same microstructure as given in Fig. 3, SCA computes the stress responses of the RVE (or any arbitrary RVEs) when a monotonically increasing strain is applied. Since SCA provides efficient evaluation of the stress state, it is convenient to train the FFNN on data made with SCA. The FFNN can then replace SCA by "learning" the stress state as a function of the strain state. This would establish a straight-forward relationship between strain and stress for near-instantaneous evaluation of RVE stress responses. Note that although a plastic material is described for the matrix material in the RVE analyzed by SCA, the FFNN described here is limited to non-linear elastic (i.e. path independent) material behavior: we approximate the plastic response with a non-linear elastic one, and focus on monotonic loading. Moreover, in the following design case presented in Sect. 4.2, the FFNN is trained to predict not only the overall RVE stress responses (as was shown in Fig. 10), but also cluster-wise, local stress responses. This is essentially the same process, but the output layer is size $N_C \times N_N(l = 1)$, with one point for each stress component for each cluster. This second form replicates the non-homogenized results of SCA.

3.4.1 Training

The training procedure for an FFNN can be reformulated as an optimization problem. We define the loss function (or cost function) as Mean Square Error (MSE) between the estimated stress and the stress computed by using SCA. Assuming one hidden layer, the optimization formulation is given by:

$$
\begin{aligned}
&find: W_{ij}^{l=2},\ b_j^{l=2},\ W_{jk}^{l=3},\ b_k^{l=3}\\
&min\ loss\ function: MSE = \frac{1}{N_T \times N_N(l=3)}\\
&\sum_{s=1}^{N_T}\sum_{k=1}^{N_N(l=3)}\left(\sigma_k^{l=3,s} - \sigma_k^{*,l=3,s}\right)^2\\
&where:\ \sigma_k^{l=3,s}\\
&= \sum_{j=1}^{N_N(l=2)} W_{jk}^{l=3}\left(\mathscr{A}\left(\sum_{i=1}^{N_N(l=1)} W_{ij}^{l=2}\varepsilon_i^{\mathrm{M},s} + b_j^{l=2}\right) + b_k^{l=3}\right)
\end{aligned}
\tag{23}
$$

By finding the optimal values for $W_{ij}^{l=2}$, $b_j^{l=2}$, $W_{jk}^{l=3}$, and $b_k^{l=3}$, MSE will be reduced. Note that only training data is used in this process, hence $s = 1, 2, \ldots, N_T$.

Usually, the MSE gradually decreases with each training step. To ensure the trained neural network is general enough for all possible input states, some data points called verification data are used to monitor trends in the error. The minimization iterations will terminate before the error of the verification data starts to increase. This will ensure the neural network is able to provide certain extrapolating capability for data points that are not within the training set.

The FFNN described in Sect. 3.3 was trained on the database described in Sect. 3.2. In this case, an FFNN with one hidden layer and 50 neurons was chosen. In the training procedure, 1000 samples are used to train the neural network. The Levenberg–Marquardt optimization algorithm is used to reduce the MSE [49,50].

3.4.2 Prediction

After the training process, a fast evaluation of the stress state during a monotonic loading process was performed. The FFNN used for this predicts the macroscale stress tensor for a given macroscale strain tensor following Eq. (24a); similarly, Eq. (24b) is used to predict the local (cluster-wise) stress tensors given a macroscale strain tensor.

$$
\boxed{\boldsymbol{\sigma}^{\mathrm{M},s(\mathrm{FFNN})} = \sigma_k^{l=N_L,s} = \mathscr{F}_{FFNN}\left(\varepsilon_i^{\mathrm{M},s}\right)} \tag{24a}
$$

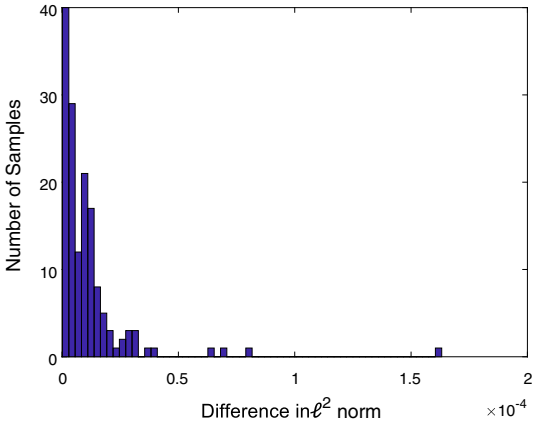

Fig. 11 Histogram of the difference between the FFNN $\boldsymbol{\sigma}^{\mathrm{M}}$ predictions and validation data set for all $N_V = 150$ samples. Most of the predictions made by FFNN has an l^2 norm less than 2×10^{-5}, showing that the FFNN produce an accurate prediction of the stress state

$$\boldsymbol{\sigma}^{s(\mathrm{FFNN})}(\boldsymbol{X}) = \sigma_k^{l=N_L,s}(\boldsymbol{X}) = \mathscr{F}_{FFNN}^{micro}\left(\varepsilon_i^{\mathrm{M},s}\right) \tag{24b}$$

To demonstrate validation of the macroscale FFFN, the l^2 norm is used to measure the difference between the overall stress predicted by Eq. (24a) and the homogenized SCA results for each sample in the validation data set, as computed by:

$$Difference_{FFNN}^{s} = \|\mathscr{F}_{FFNN}\left(\varepsilon_i^{\mathrm{M},s}\right) - \boldsymbol{\sigma}^{\mathrm{M},s(\mathrm{SCA})}, \quad s = 1, 2, \ldots, N_V, \tag{25}$$

To validate the trained FFNN, another 30 final strain states (unknown during the training) with five load steps each (including the final state) were selected. Figure 11 shows a histogram of difference measured with the l^2-norm, Eq. (25), for these new $N_V = 150$ strain–stress pairs of in the validation data set. Most of the test samples have a very low l^2-norm, which shows that the FFNN is well trained. Figure 12 shows the SCA stress data for each stress component plotted against the FFNN predictions; a perfect match has a slope of one. The associated cross-correlation statistic is one: the FFNN solutions match the SCA solutions perfectly. At each load step the stress predictions of the FFNN and SCA match, as shown in Fig. 13.

This case study illustrates a convenient workflow that used the reduced order modeling approach to generate a rich microstructure response database for training an FFNN, which is then used for generating fast predictions of the RVE responses. Note that although the validation for the FFNN for the homogenized stress-strain relationship is given in detail here, a similar process has been used for the relationship between macroscale loading and microscale (cluster-wise, or local) stresses, as given in Eq. (24b).

We propose that the FFNNs shown here can be used in a design optimization process, such as topology optimization or microstructural design, where a fast and accurate material responses prediction is desired. However, note that the material is non-linear elastic and/or under monotonic loading. If plasticity and loading/unloading are considered, a different FFNN setup or a different neural network may be required. A speed comparison of running 150 samples with SCA and FFNN is given in Table 5, where the speedup for online prediction of σ^{M} is 10,000 for the FFNN over SCA. This idea will be explored further using both the FFNN predictions (Eq. (24a) and (24b)) and convolutional neural networks in Sect. 4.

3.5 Convolutional neural network

Convolutional networks or convolutional neural networks (CNNs) are widely used in fields such as image recognition

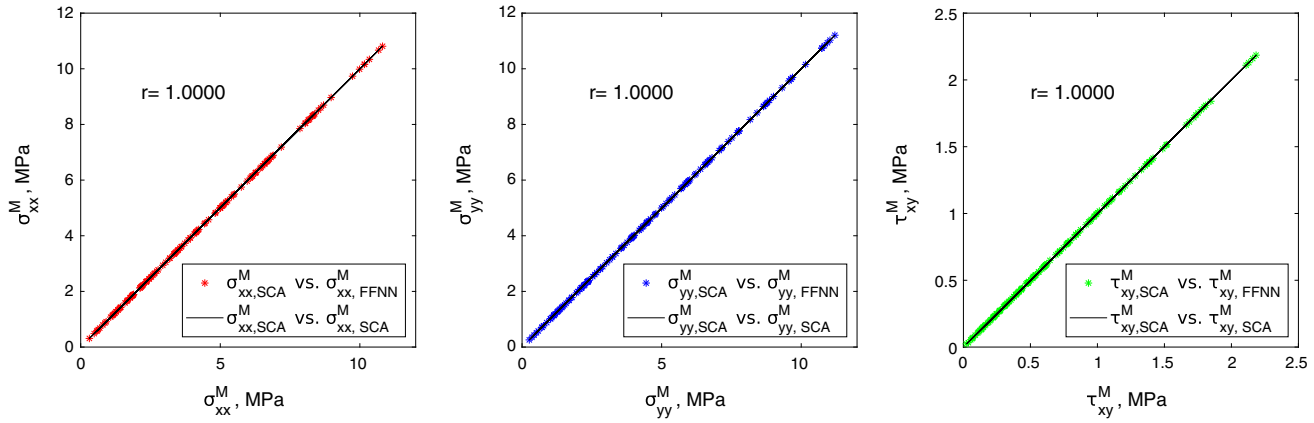

Fig. 12 Cross-correlation between SCA and FFNN for x-, y-, and shear-directions. The solid black lines are the ground truth: all perfect predictions should lay on those lines. The FFNN-predicted stress components lay around the solid black lines, and all three cases have correlation coefficients of 1, exhibiting very strong predictive confidence

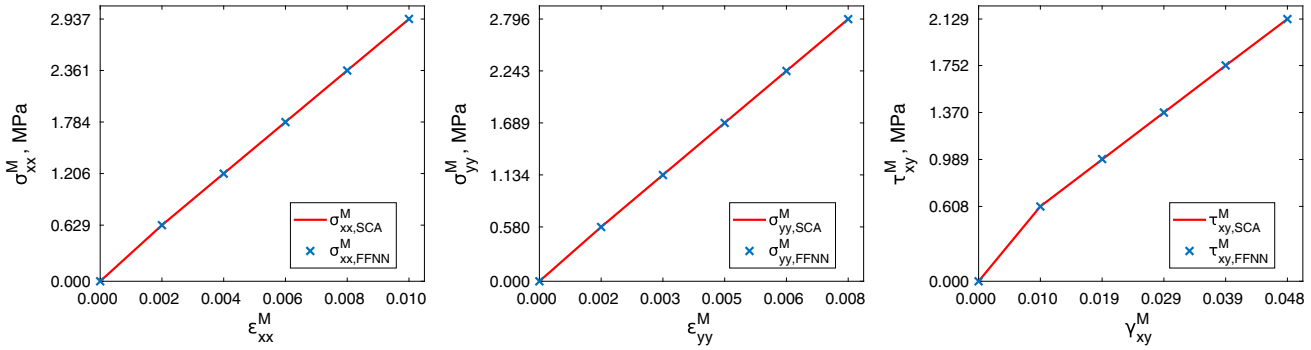

Fig. 13 Stress-strain plot with both the FFNN and SCA solutions for the validation data, showing that the FFNN results successfully reproduced SCA results

Table 5 Comparison of time required to run 150 samples using SCA and FFNN

Method	Total time (s)	Online speedup over SCA
SCA(4 + 4 clusters)	Offline: 13.0 + online: 3×10^2	–
FFNN	Training: 60 + prediction: 3×10^{-2}	10,000

and feature identification. The term "convolutional" refers to the linear mathematical operation and indicates that the convolution operation is implemented in at least one layer of the network rather than conventional matrix multiplication [51]. This is a biologically inspired model [52] used to handle known grid-like topology data such as time series (1D grid of samples at successive time intervals) or image data (2D grid of pixels). Convolution neural networks have been implemented in materials science and multiscale modeling to analyze the microstructure properties where the input data are microstructure images. Extracting material information through microstructure images, a ubiquitous data type in materials science, has proven to be a promising application of CNNs. For example, Lubbers et al. [53] implemented a CNN based on the distribution of texture images for unsupervised detection of low-dimensional structures. Scanning electron microscope (SEM) images are frequently used in materials science to distinguish between categories of materials. Such image datasets can be classified with a single feature or with multiply features using CNNs. Some studies have implemented CNN to featurize SEM images over a single set of data [54,55]. However, a scalable and a generalizable feature should be used to facilitate widespread applicability of the CNN. Ling et al. [56] analyzed the generalizability and interpretability of CNN-based featurization methods for SEM images, and found that mean texture featurization is generally useful in such cases, although sometimes feature-free CNN procedures are appealing as well [54].

The application of CNNs is not limited to images. Cang et al. [57] established a CNN approach to predict the physical properties of a heterogeneous material, replacing standard statistical or micromechanical modeling techniques. The generated scheme is applicable to systems with a highly non-linear mapping based on high dimensional microstructure. They have implemented a convolutional network to quantify material morphology followed by another convolution network to predict the material properties given the microstructure. This can also be done in 3D, for complex materials and responses [45,58,59].

A CNN model consists of several basic unit operations: padding, convolution, pooling, and a feed forward neural network (FFNN). The structure of an example 1D CNN is shown in Fig. 14.

The input is a series of stress values, i.e. a 1D problem. The 1D CNN consists of several loops of padding, convolution, and pooling. For a specific loop iteration η, a padding procedure adds zeros around boundaries, to ensure that the post-convolution dimension is the same as the input dimension. After padding, several kernel functions will be used to approximate the discrete convolution operator given by:

$$\tilde{\sigma}_x^{\kappa,\eta} = \sum_{\xi=-(L_{conv}-1)/2}^{(L_{conv}-1)/2} w_\xi^{\kappa,\eta} \sigma_{x+\xi}^{padded,\eta} + b^{\kappa,\eta}, \tag{26}$$

where $\sigma_{x+\xi}^{padded,\eta}$ is the input, $w_\xi^{\kappa,\eta}$ is the κth kernel function, and $b^{\kappa,\eta}$ is the bias, for ηth convolution process and $\eta = 1, 2, \ldots, N_{conv}$. The size of the kernel function is L_{conv}. A summary of all of the notation used in this section is given in Table 6. The convolution operation can be regarded as a feature identification operation. After padding, a pooling layer will decrease the dimension of inputs, and extract the most important features from the post-convolution data. A one dimensional max pooling equation is given by

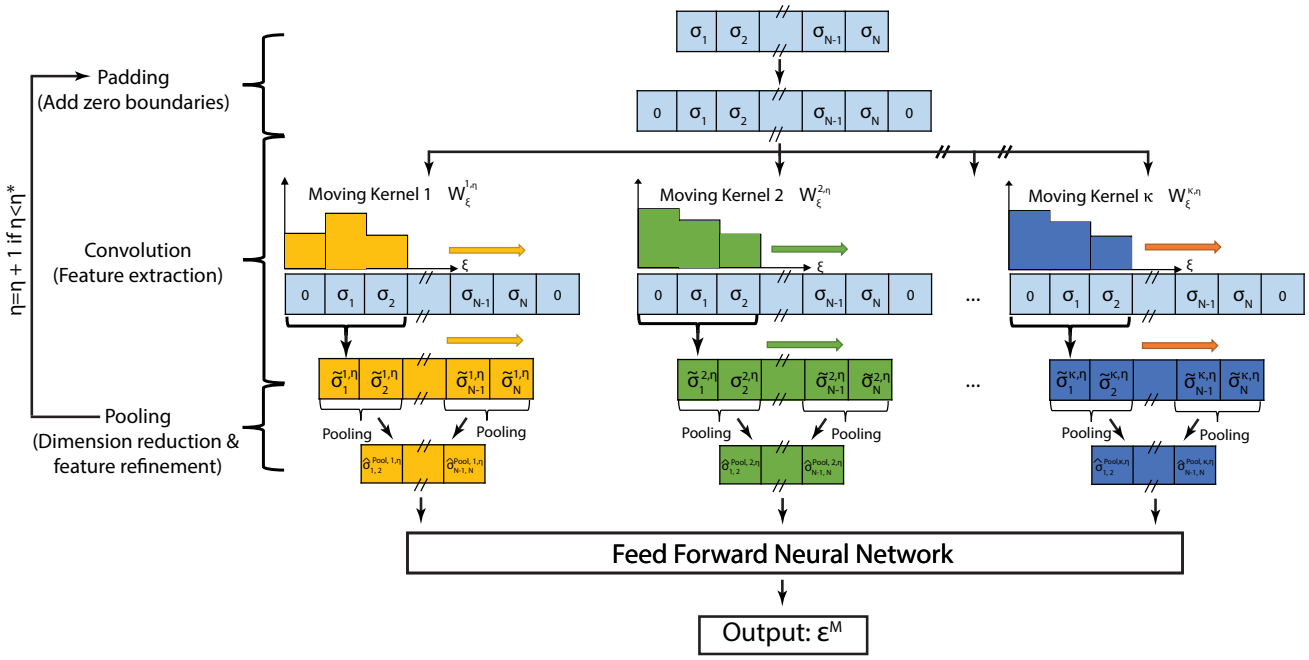

Fig. 14 Illustration of one dimensional convolutional neural network with the following setup: padding, convolution, pooling, and a feed forward neural network for regression analysis. The first three steps may be repeated

$$\hat{\sigma}_{\alpha}^{max,\kappa,\eta} = MAX\left(\tilde{\sigma}_{\xi}^{\kappa,\eta}, \xi \in [(\alpha-1)L_{pooling}+1, \alpha L_{pooling}]\right)$$
$$\alpha = 1, 2, \ldots, N_{pooling}^{\eta} \tag{27}$$

where $\hat{\sigma}^{max,\kappa,\eta}$ is the output value, $\tilde{\sigma}_{\xi}^{\kappa,\eta}$ is the input value, and $L_{pooling}$ is the length of the pooling window. Max pooling extracts the maximal value from the window, but other pooling operations might also be used. In this case we surmise that, to predict remote strains, the maximum stress values might be telling. Padding, convolution, and pooling may be repeated for N_{conv} times. The value will be transferred to a fully connected FFNN, such as that illustrated in the previous section.

Figure 15 represents a generalized overview of the structure of a typical two dimensional CNN [48]. Continuing with the two-dimensional example problem given above, for a CNN the input is the stress within the 600 × 600 grid, given by $\boldsymbol{\sigma}(\alpha, \beta)$, were α and β correspond to the x- and y-components of the stress map. At each grid point (α, β), the three stress components xx, yy and xy are stored. Padding $\mathscr{P}_{padding}$ will add zero boundaries around the input data to make sure that after convolution, the size of the maps will remain same.

Similar to 1D convolution, the convolution operation applies a kernel function over the stress contours and generates a new feature map with the same resolution as the initial stress contours. The extension of the convolution operation to two dimensions for ηth convolution process is shown in Eq. (28).

$$\tilde{\sigma}_{\alpha,\beta}^{\kappa,\eta} = \sum_{\zeta=1}^{N_{feature}} \times \left(\sum_{\xi=-(X_{conv}-1)/2}^{(X_{conv}-1)/2} \sum_{\psi=-(Y_{conv}-1)/2}^{(Y_{conv}-1)/2} W_{\xi,\psi}^{\zeta,\kappa,\eta} \sigma_{\alpha+\xi,\,\beta+\psi}^{padded,\zeta,\eta} \right) + b^{\kappa,\eta} \tag{28}$$

where κ is the kernel ID of the convolution layer goes from one to N_{kernel}. The size of the convolution kernel in dimension 1 and 2 are given by X_{conv} and Y_{conv}, and are both odd numbers. The counting indices in the kernel in dimension 1 and 2 are defined as ξ and ψ, respectively. The number of stress components is $N_{feature}$; for this 2D example, $N_{feature}$=3. By applying the kernel to each element in each the input stress array, a complete feature map will be generated.

To define a nonlinear relationship between the input and output using the CNN, a nonlinear activation function is often used. In some cases, this is a Rectified Linear Unit (ReLU) layer, which is applied to all feature maps generated from the convolution operation. For simplicity of illustration, this step is not shown in the equations and figures.

A pooling layer is applied to all feature maps after the ReLU layer to compress the resolution of the data in the X and Y directions. Different pooling operations might be used; in this example, we selected max pooling. The max pooling operator divides the feature map into many subset regions, and selects the maximum value from each region to use as the value in the compressed feature map; generalizing from

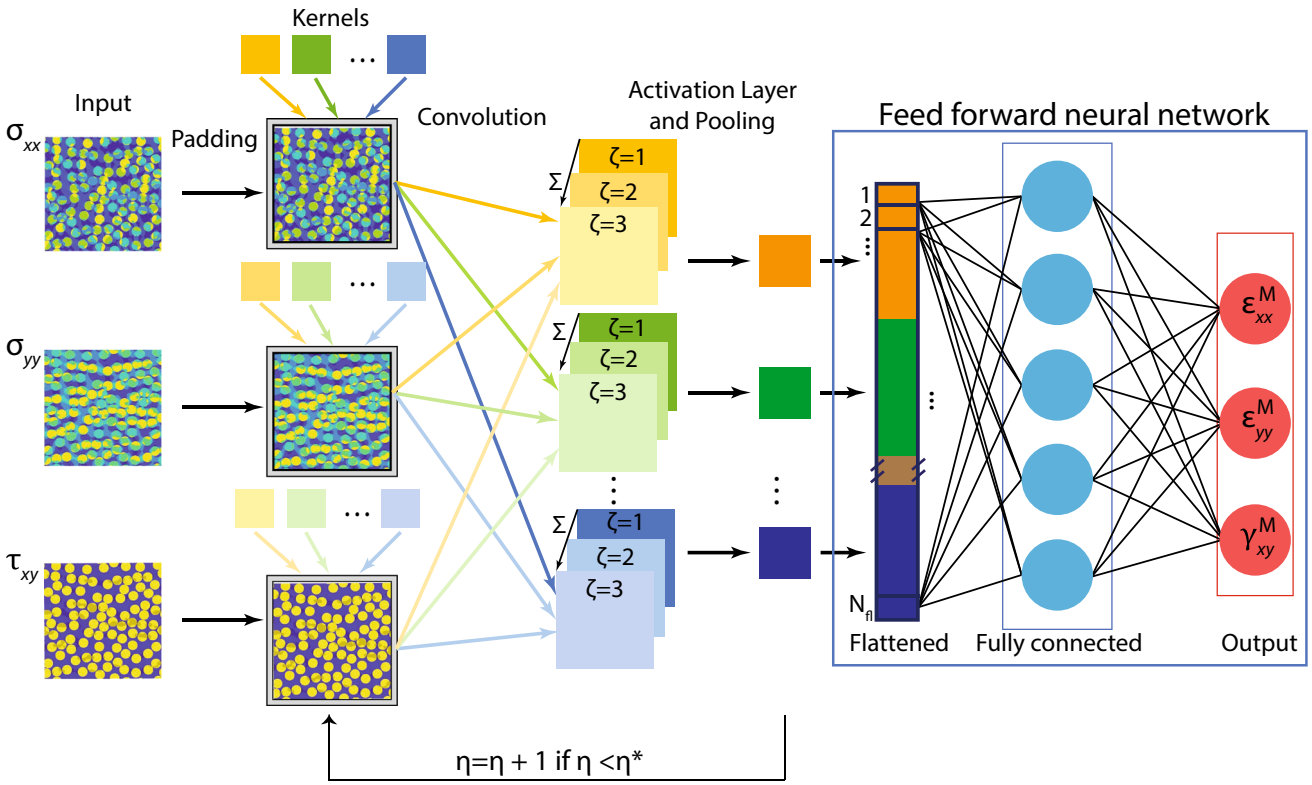

Fig. 15 Illustration of two dimensional convolutional neural network. In the training part, the number of threshold $\eta^* = 5$. After 4 repeats of convolution, the data will be passed to FFNN. The dimension of input for FFNN becomes $N_{fl} = 74$. The hidden layer in FFNN has 74 neurons. Finally, FFNN gives three macro strain components

Eq. (27), for ηth convolution process, this can be written as:

$$\begin{aligned}\hat{\sigma}^{max,\kappa,\eta}_{\alpha,\beta} &= MAX(\tilde{\sigma}^{\kappa,\eta}_{\xi,\psi},\\ &\xi \in [(\alpha-1)X_{pooling}+1, \alpha X_{pooling}],\\ &\psi \in [(\beta-1)Y_{pooling}+1, \beta Y_{pooling}])\\ &\alpha = 1,2,\ldots,N^{\eta}_{Xpooling},\ \beta = 1,2,\ldots,N^{\eta}_{Ypooling}\end{aligned} \tag{29}$$

After the final pooling operation, all compressed feature maps are converted into a single vector through a flattening operation. The flattened array is then used as the input of the FFNN for regression to compute the corresponding strain. Further details of the CNN method and implementation can be found in literature cited in the beginning of this section.

3.6 Convolutional neural network for boundary condition identification with database generated by SCA

Using the CNN illustrated in Fig. 15, a mapping has been established between local stress distribution and applied external strain on the microstructure.

3.6.1 Training

This CNN was trained on the same database of 1000 SCA results as was the FFNN. For the CNN, the input is the micro-scale stress at each point (voxel) of the RVE, given by the cluster-wise results of SCA, $\boldsymbol{\sigma}(\alpha, \beta)$. Each point in the RVE contains the three 2D stress components, like the RGB channels used for images. The output of the CNN is the macroscale strain $\boldsymbol{\varepsilon}^M$ that was applied as the loading conditions and caused the observed stresses. The training can be written as an optimization problem, as given in Eq. (30). The equations for training a CNN with padding, convolution, and pooling layers repeated N_{conv} times is given by:

$$\begin{aligned}&find: W^l_{mn},\ b^l_n\ (l = 2,3...N_L),\ \text{in FFNN}\\ &W^{\zeta,\kappa,\eta}_{\xi,\psi}, b^{\kappa,\eta}(\kappa = 1,2...N_{kernel}), (\eta = 1,2...N_{conv}),\ \text{in CNN}\\ &min\ loss\ function: MSE = \frac{1}{N_T}\sum_{s=1}^{N_T}\left(\boldsymbol{\varepsilon}^{M,s} - \boldsymbol{\varepsilon}^{*M,s}\right)^2\\ &where: \boldsymbol{\varepsilon}^{M,s}\\ &= \mathscr{F}_{FFNN}\left(\mathscr{F}_{flatten}\left(\mathscr{F}^{N_{conv}}_{pcp}\left(\ldots\mathscr{F}^{2}_{pcp}\left(\mathscr{F}^{1}_{pcp}\left(\boldsymbol{\sigma}^{s}(\alpha,\beta)\right)\right)\ldots\right)\right)\right)\end{aligned} \tag{30}$$

Table 6 Notation used to describe the CNNs shown in Sect. 3.5 through Sect. 3.7

$\boldsymbol{X}$	A point in the microscale (inside the RVE) region
$\boldsymbol{\sigma}^{\mathrm{M}}$	Macroscale stress tensor
$\boldsymbol{\varepsilon}^{\mathrm{M}}$	Macroscale strain tensor
$\boldsymbol{\sigma}(\boldsymbol{X})$	Microscale stress tensor
$\boldsymbol{\varepsilon}(\boldsymbol{X})$	Microscale strain tensor
α	First direction in $\boldsymbol{X}$
β	Second direction in $\boldsymbol{X}$
$\tilde{\boldsymbol{\sigma}}$	Microscale stress tensor to which a kernel has been applied
$\hat{\boldsymbol{\sigma}}$	Microscale stress tensor to which a kernel and pooling has been applied
N_T	Number of training samples in the database
N_V	Number of verification samples in the database
N_{kernel}	Number of kernels in convolution
$N_{pooling}, N_{Xpooling}, N_{Ypooling}$	Size of output after pooling
N_{fl}	Number of entries in flattened vector
N_{conv}	Number of repeats of padding, convolution, and pooling in CNN
W_{ij}^{l}	Weight in FFNN connecting the ith neuron in layer $l-1$ to the jth in layer l
b_j^l	Bias in FFNN of the jth neuron in layer l
ξ	Counting index for location within kernel in dimension 1
ψ	Counting index for location within kernel in dimension 2
ζ	Counting index for features
κ	Counting index for kernels
η	Counting index for convolutions
$W_{\xi,\psi}^{\zeta,\kappa,\eta}$	Weight connecting the input for ζ feature and κ kernel on convolution layer
$b^{\kappa,\eta}$	Bias of the kernel κ on the convolution layer
$L_{conv}, X_{conv}, Y_{conv}$	Size of the kernel
$L_{pooling}, X_{pooling}, Y_{pooling}$	Size of the pooling window
$\mathscr{P}_{padding}$	Padding function
$\mathscr{C}_{conv}$	Convolution function
$\mathscr{P}_{pooling}$	Pooling function
$\mathscr{F}_{pcp}$	Combined padding, convolution, and pooling operation
$\mathscr{F}_{flatten}$	Flattening function
$\mathscr{F}_{CNN}$	Convolutional neural network function
$\mathscr{F}_{CNN}^{classify}$	Classification convolutional neural network function
d	Binary indicator given by classification CNN

The term $\mathscr{F}_{pcp}^{\eta}\left(\boldsymbol{\sigma}^{s}(\alpha,\beta)\right)$ is used to simplify the notation by combining the nested operations shown in Fig. 15. It is defined as

$$\mathscr{F}_{pcp}^{\eta}\left(\boldsymbol{\sigma}^{s}(\alpha,\beta)\right)=\mathscr{P}_{pooling}^{\eta}\left(\mathscr{C}_{conv}^{\eta}\left(\mathscr{P}_{padding}^{\eta}\left(\boldsymbol{\sigma}^{s}(\alpha,\beta)\right)\right)\right),$$

which includes terms for padding, convolution, and pooling. The remaining terms in the training problem are define as follows. The weight and bias in the FFNN are W_{mn}^{l} and b_m^l, and $W_{\xi,\psi}^{\zeta,\kappa,\eta}$, $b^{\kappa,\eta}$ are the weights and biases in the convolution operations. The ground truth is $\boldsymbol{\varepsilon}^{*M,s}$ and the estimate is $\boldsymbol{\varepsilon}^{M,s}$. The number of training samples is defined as N_T, and $N_N(l=N_L)$ is the number of neurons in the output layer. In this case, $N_N(l=N_L)$ is three. The sampling is indexed by s.

The inputs to the CNN are the three stress arrays corresponding to the components of stress given by $\boldsymbol{\sigma}^{s}(\alpha,\beta)$. The outputs are the three strain values $\varepsilon_j^{M,s}(j=1,2,3)$. Just as for the FFNN, the mean squared error (MSE) is reduced gradually step by step using one of several optimization algorithms.

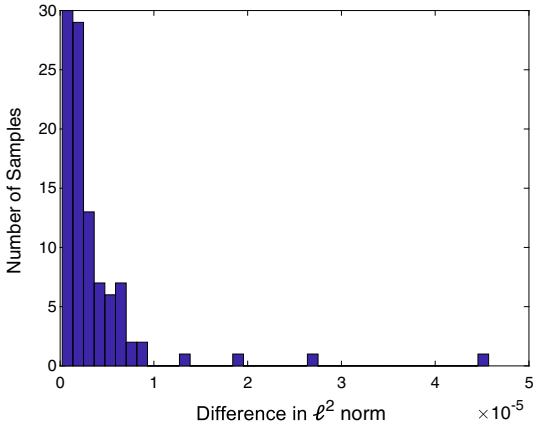

Fig. 16 Histogram of the l^2 norm for the CNN prediction using validation data points. Most of the predictions made by CNN has an l^2 norm less than 1×10^{-5}, showing the CNN produce an accurate prediction of the strain state. The l^2 norm illustrate the CNN network is able to make a proper prediction of the validation data

3.6.2 Prediction

Just as with the FFNN, we can define the function $\mathscr{F}_{CNN}(\boldsymbol{\sigma}^s(\alpha, \beta))$ that describes the operation performed by the trained CNN, in this case

$$\boxed{\mathscr{F}_{CNN}\left(\boldsymbol{\sigma}^s(\alpha, \beta)\right) = \boldsymbol{\varepsilon}^{\mathrm{M},s}}. \tag{31}$$

In Fig. 16, a histogram for the error between predicted $\boldsymbol{\varepsilon}^{\mathrm{M}}$ and the validation data set (again, the $N_T = 150$ samples generated in Sect. 3.2) is computed using Eq. (32).

$$Difference^s_{CNN} = \|\mathscr{F}_{CNN}\left(\boldsymbol{\sigma}^s(\alpha, \beta)\right) - \boldsymbol{\varepsilon}^{\mathrm{M},s(\mathrm{SCA})}\|_2., \quad s = 1, 2, \ldots, N_V \tag{32}$$

In Fig. 17, the correlations between the CNN prediction of applied strain and the reference solution of the three strain components is provided, using the same validation data sets as mentioned before. These show that the CNN can effectively map the stress contour to the applied external strain. Such a map may play an important role in linking microstructure information with macroscale information, e.g. connecting microstructure failure strength to a macroscale strain state. This will assist the inverse design problem where the optimum loading state is inferred using local information.

3.7 Convolutional neural network for classification with database generated by SCA

The application of a CNN to material microstructure predictions is not limited to the sample problem shown above. Another example use of a CNN is as a microstructure classifier; this is similar to its common application in image classification. By using a microstructure mesh and an applied strain on the microstructure as the input, a CNN can be trained to predict whether the microstructure will become damaged.

3.7.1 Training

The training procedure for the classification CNN is given by

$$\begin{aligned}
&find: W^l_{mn},\ b^l_n\ (l = 2, 3...N_L),\ \text{in FFNN}\\
&\quad W^{\zeta,\kappa,\eta}_{\xi,\psi}, b^{\kappa,\eta}(\kappa = 1, 2...N_{kernel}), (\eta = 1, 2...N_{conv}),\ \text{in CNN}\\
&min\ loss\ function: cross\ entropy\\
&= \frac{1}{N_T}\sum_{s=1}^{N_T}\left(-\left(d^{*s}\log\left(d^s\right) + \left(1 - d^{*s}\right)\log\left(1 - d^s\right)\right)\right)\\
&where: d^s\\
&= \mathscr{F}_{FFNN}\left(\mathscr{F}_{flatten}\left(\mathscr{F}^{N_{conv}}_{pcp}\left(...\mathscr{F}^2_{pcp}\left(\mathscr{F}^1_{pcp}\left(\boldsymbol{\sigma}^s(\alpha, \beta)\right)\right)...\right)\right)\right)\\
&nested\ operation: \mathscr{F}^{\eta}_{pcp}\left(\boldsymbol{\sigma}^s(\alpha, \beta)\right)\\
&= \mathscr{P}^{\eta}_{pooling}\left(\mathscr{C}^{\eta}_{conv}\left(\mathscr{P}^{\eta}_{padding}\left(\boldsymbol{\sigma}^s(\alpha, \beta)\right)\right)\right)
\end{aligned} \tag{33}$$

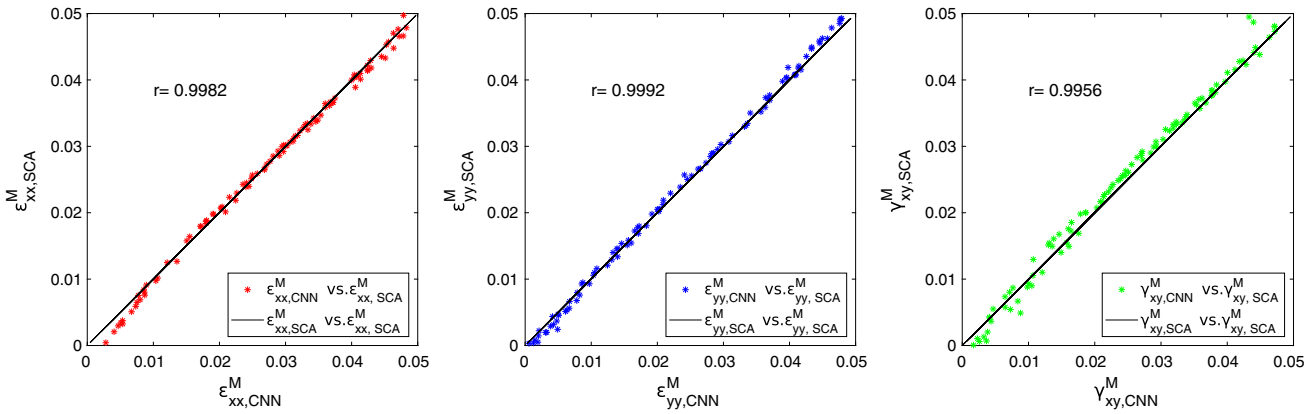

Fig. 17 Loading prediction by CNN vs. loading applied in SCA using validation data points. The solid black lines are the ground truth: all perfect predictions should lay on those lines. All three cases have correlation coefficients higher than 0.99, suggesting the trained CNN can provide a good accuracy in predicting applied strain

The output of the classification CNN, d^s, is a binary indicator: 0 for non-damaged, 1 for damaged. Since the output is a binary value, the objective function is now defined in terms of the cross entropy between the truth value d^{*s} and the predicted value d^s. Cross entropy, or log loss, is widely used to measure the performance of a classification model [22]. The CNN in this section is trained on the data extracted from the database used for the previous NNs. The database for training the classifier consists of pairs of micro stress distributions and damage indicators. In this case, a critical-stress-based damage criterion is used to decide whether an RVE is damaged or not: if any von Mises stress $\sigma(\boldsymbol{X})$ exceeds a critical von Mises stress σ^* the RVE is considered damaged.

3.7.2 Prediction

Once trained, the CNN can predict whether the applied loading will result in microstructure damage without carrying out the full RVE simulation:

$$\mathscr{F}_{CNN}^{classify}\left(\boldsymbol{\sigma}^s(\alpha,\beta)\right) = d^s = \begin{cases} 0, & \text{Non-damaged} \\ 1, & \text{Damaged} \end{cases}. \quad (34)$$

In Sect. 4.2, such a CNN will be used in a microstructure-based topology optimization example to illustrate the effect of a microstructural damage constraint on the optimized structure.

4 Microstructure-based, multiscale topology optimization using neural networks

In this section, we will illustrate how FFNNs and CNNs might be used in topology optimization to achieve microstructure-based design. This differentiates the current approach from classical topology optimization which typically uses simple constitutive relationships. As explained in Sect. 3, we propose to compress the RVE response database into an FFNN for forward prediction of RVE stress responses, where it will act similarly to a traditional homogenized constitutive model. However, because a database of RVE responses is used, no functional form of the homogenization is required. The RVE microstructure damage responses is represented with a trained FFNN+CNN; this introduces microstructure damage, linked with the applied strain state of the RVE. By using well-trained FFNNs and CNNs, two different optimization problems are defined as below:

(a) Topology optimization with a material constitutive law extracted from an FFNN trained on the stress-strain relationship of a given microstructure. In this case, a non-linear material behavior during topology optimization is used to achieve a design that is durable under extreme loading conditions where the material response enters the non-linear region.
(b) Topology optimization with constraints defined by the FFNN+CNN framework to identify microstructure damage and thereby design durable (damage aware) structures, using the CNN described in Sect. 3.7. In this case, the microstructure damage acts as an extra constraint to the topology optimization formulation to achieve a design that alleviates or avoids possible local microstructure damage.

In these two example problems, the design zone is described with a 60×30 mesh of rectangular, linear elements. Each element is a $1\,\text{cm} \times 1\,\text{cm}$ square. The elastic material properties are from the homogenized SCA results: $E = 200\,\text{MPa}$ and $\nu = 0.27$. An approximated non-linear elastic material response is extracted from the RVE simulation obtained from Sect. 3, as described in the following sections. Note that while plasticity is used in Sect. 3, our FFNN is only valid for a non-linear elastic approximation of the material response. A point load of 75 N is applied at right bottom corner. The desired volume fraction of the optimized part is set as 0.35 of the original design zone. For the second case with damage the critical stress is defined as 0.7 MPa.

4.1 Topology optimization with FFNN

The formulation of microstructure sensitive topology optimization with an FFNN is shown in Eq. (35). The objective function is defined as the overall strain energy of the structure, Φ, which is to be minimized. A subtle but important difference in the present example in this work is that the FFNN is used to replace the usual linear elastic material response with a nonlinear one. By using a nonlinear material responses database depicted in Sect. 3.3, a new avenue for data-driven material and structure design is illustrated. This is depicted graphically in Fig. 18, which shows the use of an FFNN to generate microstructure-based stress-strain response within a topology design framework.

$$\begin{aligned}
&minimize: \ \Phi = \int_{\Omega^M} \boldsymbol{f}\boldsymbol{u}\mathrm{d}\Omega^M + \int_{\partial\Omega^M} \boldsymbol{t}\boldsymbol{u}\mathrm{d}S^M, \forall \boldsymbol{u} \in U^* \\
&with: \ \Phi \equiv \int_{\Omega^M} \boldsymbol{\sigma}(\boldsymbol{X}^M)\boldsymbol{\varepsilon}(\boldsymbol{X}^M)\mathrm{d}\Omega^M, \forall \boldsymbol{X}^M \in \Omega^M \\
&subject\ to: \ V\left(\rho(\boldsymbol{X}^M)\right) \le V^* \\
&\quad 0 \le \rho(\boldsymbol{X}^M) \le 1, \forall \boldsymbol{X}^M \in \Omega^M \\
&\quad \boldsymbol{\sigma}(\boldsymbol{X}^M) = \rho(\boldsymbol{X}^M)\mathscr{F}_{FFNN}\left(\boldsymbol{\varepsilon}(\boldsymbol{X}^M)\right), \forall \boldsymbol{X}^M \in \Omega^M \ \leftarrow \text{FFNNfor} \\
&\quad \text{microstructural response, Eq.(24a)} \\
&where: \ V(\rho(\boldsymbol{X}^M)) = \frac{1}{\Omega^M}\int_{\Omega^M} \rho(\boldsymbol{X}^M)\mathrm{d}\boldsymbol{X}^M \\
&\Omega^M: \text{ macro domain, } \Omega: \text{ micro domain} \\
&\boldsymbol{X}^M: \text{ coordinate in macro domain, } \boldsymbol{X}: \text{ coordinate in micro domain}
\end{aligned} \quad (35)$$

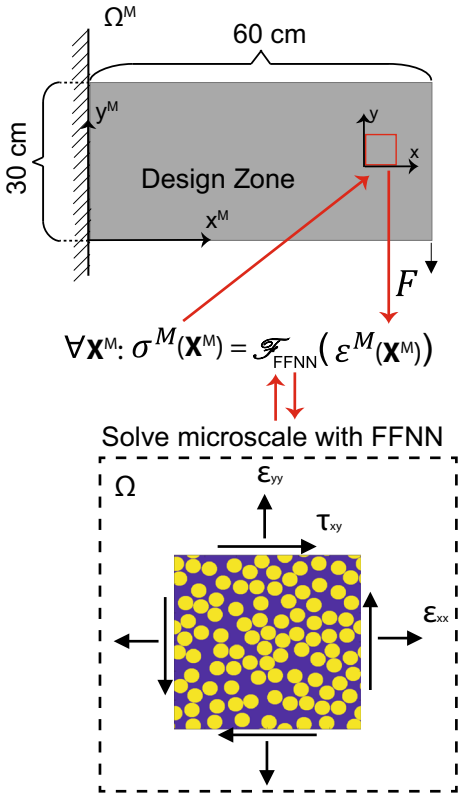

Fig. 18 Topology optimization setup with FFNN. The FFNN will be used to compute non-linear material responses to drive for a new design. This replaces the constitutive law commonly used for the macroscale with a homogenized response of the microstructure for each point in the macroscale. Mathematically, $\boldsymbol{\sigma}^M(\boldsymbol{X}^M) = \rho(\boldsymbol{X}^M)\mathscr{F}_{FFNN}\left(\boldsymbol{\varepsilon}(\boldsymbol{X}^M)\right), \forall \boldsymbol{X}^M \in \Omega^M$, as defined in Eq. (35)

where $\boldsymbol{f}$ is the body force, $\boldsymbol{t}$ is the applied traction on the boundary of the design zone, and $\boldsymbol{u}$ is the local displacement in the design zone. U^* is the admissible displacement field, and S^M defines the boundaries of the macrostructure (the design region). $\boldsymbol{\sigma}(\boldsymbol{X}^M)$ and $\boldsymbol{\sigma}(\boldsymbol{X}^M)$ are macro stress and strain. The density for each macro mesh is $\rho(\boldsymbol{X}^M)$. The desired volume of material remaining in the design zone is V^* and defined as 0.35, and $V\left(\rho(\boldsymbol{X}^M)\right)$ is the optimized volume in the design zone. The density of location $\boldsymbol{X}^M$ is $\rho(\boldsymbol{X}^M)$ in the design zone, and the homogenized stress is $\boldsymbol{\sigma}(\boldsymbol{X}^M)$, defined as the product of $\rho(\boldsymbol{X}^M)$ and $\mathscr{F}_{FFNN}\left(\boldsymbol{\varepsilon}(\boldsymbol{X}^M)\right)$. Here, $\mathscr{F}_{FFNN}$ represents a trained FFNN that will generate stress predictions based on given strain input $\boldsymbol{\varepsilon}(\boldsymbol{X}^M)$, as defined in Eq. (17)

In this case, the FFNN is used to approximate the RVE responses. Hence, $\boldsymbol{\sigma}^M$ is directly approximated by the FFNN. In truth, constraints of the software used for optimization require a functional description of the behavior (this limitation will be addressed in future work), thus the effective von Mises RVE stress versus effective strain curve is approximated using an exponential function: $\bar{\sigma}^M = 0.7784 * e^{22.18*\bar{\varepsilon}^M} - 0.8071 * e^{-378*\bar{\varepsilon}^M}$. This hyperelastic material definition was fit to the FFNN results. This ad-hoc approach ensures stability of the optimization process by minimizing the overall strain energy using condition-based optimization [60].

The optimized beam structures are illustrated in Fig. 19a and 19b for linear elastic material and non-linear elastic material, respectively. Necessary results are provided in Table 7. The two structures show substantial difference in the final shape of the structure. This means the material non-linearity plays an important role in topology optimization, where a new truss structure is realized in order to ensure minimal strain energy. The result suggests the importance of considerable of microstructure-based material non-linearity into structure optimization. It may also be possible to include microstructure variation, such as different particle volume fractions, into the structure. In short, FFNN provides an alternative for a data-driven microstructure-based topology optimization, where the microstructural effect can be incorporated into the process to achieve different designs that meet the design criteria.

4.2 Topology optimization with constraints defined by FFNN+CNN

Topology optimization results may have high stress concentration zones. If not treated properly, the concentration zones may cause unexpected damage and affect the function of the structure. A traditional way to address this is to add stress or strain constraints to optimization. Previous authors have focused on optimization with stress concentration and singularities, e.g. [61–64]. In these previous studies, most damage criteria are only related to the macroscopic material model and do not consider microstructure mechanical behavior. In this work, an FFNN+CNN framework trained by the database generated with SCA in Sect. 3 provides a microstructure-based prediction of damage as the damage criterion.

In order to do this, the FFNN defined in Eq. (24b) that performs the operation $\boldsymbol{\sigma}(\boldsymbol{X}) = \mathscr{F}^{micro}_{FFNN}\left(\boldsymbol{\varepsilon}^M\right)$ is used. As mentioned in Sect. 3, this predicts local, rather than homogenized, stresses in the RVE. These local stress distributions serve as input to the CNN outlined in Sect. 3.7, which indicates whether or not damage has occurred. The optimization formulations are thus:

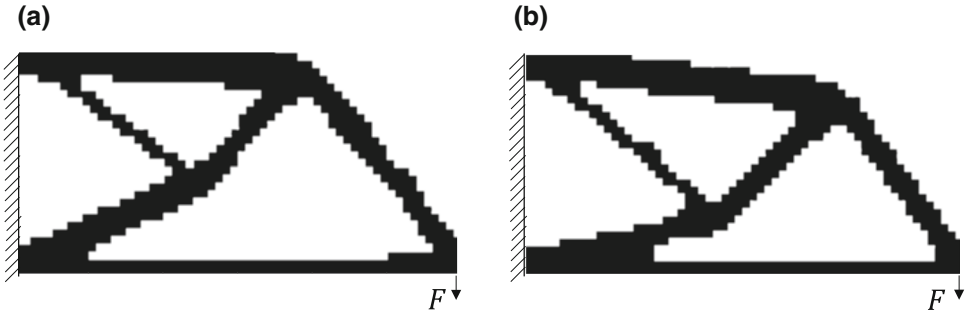

Fig. 19 **a** Optimized beam structure with elastic material responses **b** Optimized beam structure with non-linear material responses. Comparing **a** and **b**, the joint of all truss members for the non-linear case is located towards the bottom side of the truss structure. This means the material non-linear responses plays an important role in determining optimized truss structure

Table 7 Results of the FFNN-based non-linear-elastic optimization problem

	Linear material	FFNN (nonlinear)	FE-SCA concurrent[a]	FE–FE concurrent[a]
Initial compliance (strain energy) (N cm)	12.6 (6.3)	20.0 (10.0)	–	–
Optimized compliance (strain energy) (N cm)	28.0 (14.0)	38.0 (19.0)	–	–
Database generation + training (s)	0	313	–	–
Optimization calculation time (s)	338	472	23,328	220×10^6
Factor of speed-up over FE–FE	–	280,255	9431	–
No. of iterations	14	18	–	–

[a]Estimated, assuming the same number of iterations as FFNN. Note that substantial speedup is achieved while retaining the accuracy and microstructural basis of the concurrent approaches. This would provide further speed advantages in 3D

$$
\begin{aligned}
&minimize: \ \Phi = \int_{\Omega^M} \boldsymbol{f}\boldsymbol{u}\mathrm{d}\Omega^M + \int_{\partial\Omega^M} \boldsymbol{t}\boldsymbol{u}\mathrm{d}S^M, \forall u \in U^* \\
&\quad \equiv \boldsymbol{U}^T\boldsymbol{K}\boldsymbol{U} = \sum_{e=1}^{N} (\rho(\boldsymbol{X}^M))^p \boldsymbol{u}_e^T \boldsymbol{k}_0 \boldsymbol{u}_e \\
&subject\ to: \ V\left(\rho(\boldsymbol{X}^M)\right) \le V^* \\
&\quad 0 \le \rho(\boldsymbol{X}^M) \le 1 \ \forall \boldsymbol{X}^M \in \Omega^M \\
&\quad \forall \boldsymbol{X}^M \in \Omega^M(\boldsymbol{X}^M), \\
&\quad \forall \boldsymbol{X} \in \Omega(\boldsymbol{X}) : \mathscr{F}_{CNN}^{classify}(\boldsymbol{\sigma}(\boldsymbol{X})) = 0 \\
&\quad \leftarrow \text{Microscale damage criterion, Eq.(34)} \\
&where: \boldsymbol{\sigma}(\boldsymbol{X}) = \mathscr{F}_{FFNN}^{micro}\left(\boldsymbol{\varepsilon}^M\right) \\
&\quad \leftarrow \text{Microscale stress prediction, Eq.(24b)} \\
&\quad \Omega^M: \text{ macro domain, } \Omega: \text{ micro domain} \\
&\quad \boldsymbol{X}^M: \text{ coordinate in macro domain,} \\
&\quad \boldsymbol{X}: \text{coordinate in micro domain} \qquad (36)
\end{aligned}
$$

where $\Phi = \sum_{e=1}^{N} (\rho(\boldsymbol{X}^M))^p \boldsymbol{u}_e^T \boldsymbol{k}_0 \boldsymbol{u}_e$ is the compliance of the overall structure, N is the number of elements, p is the penalization power (typically $p = 3$), $\boldsymbol{u}_e$ is the displacement for each element, and $\boldsymbol{k}_0$ is the element stiffness. The averaged strain response of the RVE is $\boldsymbol{\varepsilon}^M$. The local stresses within the RVE are defined as $\boldsymbol{\sigma}$, and are calculated with a trained FFNN $\mathscr{F}_{FFNN}^{micro}$. Other variables are the same as defined above for the FFNN-only optimization. For any $\boldsymbol{X}$ in RVE $\Omega^m(\boldsymbol{X})$, the output value of $\mathscr{F}_{CNN}^{classify}$ should be 0, which represents a non-damaged state.

Since the FFNN+CNN database only gives a criterion, sensitivity analysis is not preferred in this case. The algorithm here follows a refined optimally criteria (OC) method. The optimization program structure is based on the 99-line topology optimization code written by Sigmund [65] (and thus the problem is similar, though the implementation is not identical to the FFNN-based optimization above). During the density update, for each element, three strain components will be passed to the FFNN+CNN. The FFNN+CNN will determine whether the current strain state is acceptable by assessing the local microstructural response. This is shown schematically in Fig. 20. If an element has been damaged, the density of this element will be increased by applying a penalty factor, while the densities of the rest of the elements will be decreased to satisfy the volume constraint.

Similarly to the FFNN example, Fig. 21a shows the reference case: a linear elastic optimization without a damage constraint. The final compliance is 30 N cm. Figure 21b shows the optimized design with a microstructure-based damage constraint. The final compliance is 31 N cm. Notice that while the compliance is quite similar, the design under a microstructural damage constraint resembles a more con-

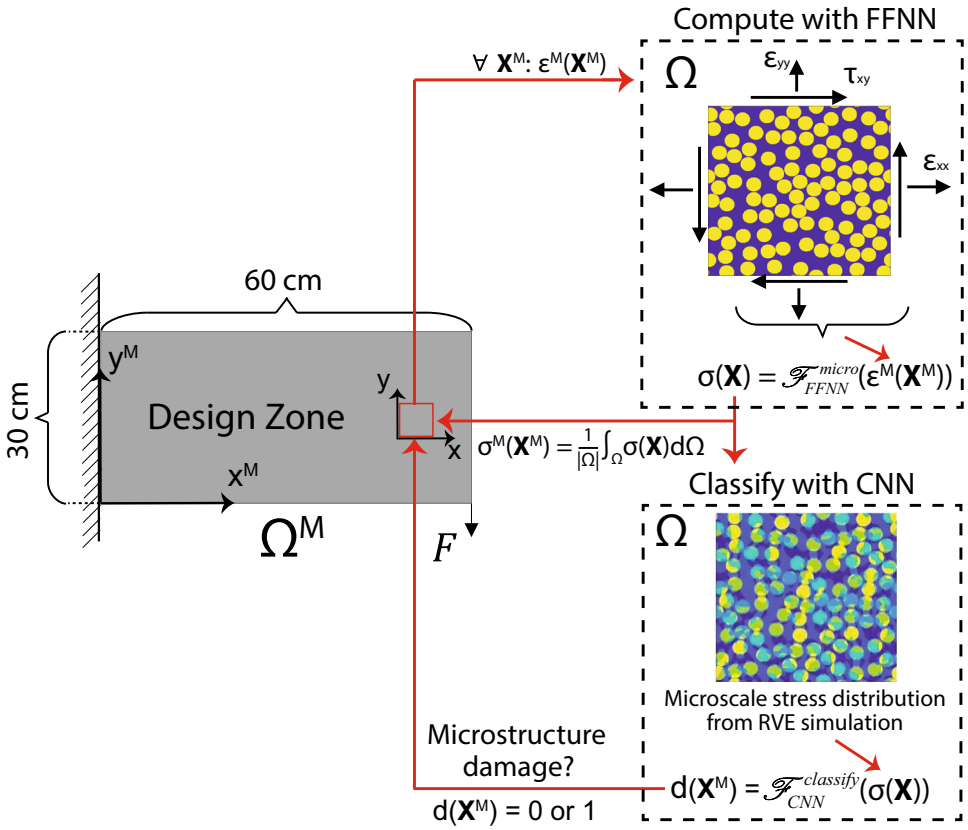

Fig. 20 Topology optimization setup with FFNN and CNN. For each material point within the design zone, the FFNN is used to compute the material response, be it linear or non-linear, considering the effect of microstructure. The CNN is used to incorporate microstructure damage, which will drive the optimization algorithm for a new design compared to a topology optimization with only linear material. Mathematically, this is $\forall \boldsymbol{X}^M \in \Omega^M(\boldsymbol{X}^M)$: $\boldsymbol{\sigma}(\boldsymbol{X}) = \mathscr{F}^{micro}_{FFNN}\left(\boldsymbol{\varepsilon}^M(\boldsymbol{X}^M)\right)$, $\forall \boldsymbol{X} \in \Omega(\boldsymbol{X})$: $d\left(\boldsymbol{X}^M\right) = \left(\mathscr{F}^{classify}_{CNN}\left(\boldsymbol{\sigma}(\boldsymbol{X})\right)\right)$, as defined in Eq. (36). If any $d(\boldsymbol{X})$ is marked as damaged in the microstructure, the $\boldsymbol{X}^M$ point in the design zone containing that microstructure will be marked as damaged

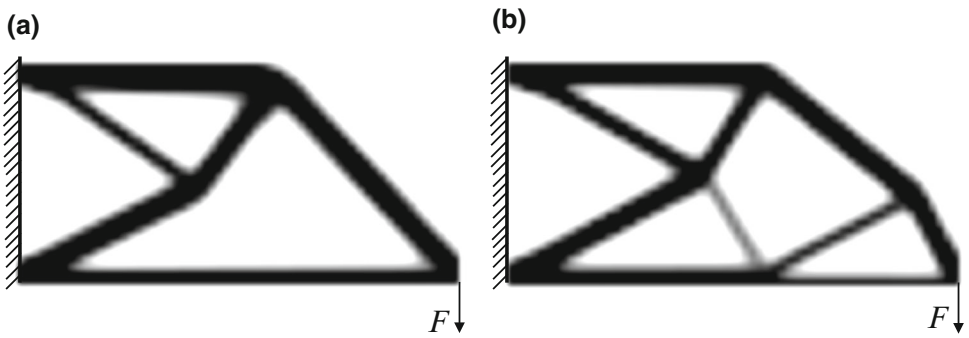

Fig. 21 **a** Optimized beam structure without damage constraints **b** Optimized beam structure with damage constraints. In **b** there are more trusses to avoid high local stresses that result in damage

ventional truss structure; the optimization has avoided sharp angles and has fewer beams that give rise to stress concentrations likely to result in microstructure-driven damage.

A summary of the design variables and important parameters related to the optimization is provided in Table 8. The simple examples above show the potential application of an FFNN+CNN database generated by clustering reduce order methods. However, the optimization is just based on an artificially defined optimality criterion. The algorithm may not be stable for all kinds of problems. In the future, we should study the sensitivity and singularity of constraints based on an FFNN+CNN database.

5 Conclusion

5.1 Summary

Two challenges with current approaches to machine learning methods in the mechanical science of materials are: (1) the database generation time and effort are extensive, and (2) the application of machine learning is not well developed or understood by the community. This paper covers several different topics related to these challenges:

– We have outlined, related, and compared three different clustering-discretization methods (SCA, VCA, and FCA)

Table 8 Results of the FFNN+CNN constraint optimization problem

	Linear material	FFNN+CNN	FE-SCA concurrent[a]	FE–FE concurrent[a]
Initial compliance (strain energy) (N cm)	295 (148)	295 (148)	–	–
Optimized compliance (strain energy) (N cm)	30.0 (15.0)	31.0 (15.5)	–	–
Database generation + training time(s)	0	512	–	–
Optimization calculation time (s)	12.6	14.5	69,674	660×10^6
Factor of speed-up over FE–FE	–	45×10^6	9473	–
No. of iterations	53	61	–	–

[a]Estimated, assuming the same number of iterations as CNN. Note that substantial speedup is achieved while retaining the accuracy and microstructural basis of the concurrent approaches. This would provide further speed advantages in 3D

that rely on unsupervised learning for order reduction and the solution of mechanistic governing equations for prediction.
- One of these methods, SCA, was used to develop an example material behavior database suitable for training neural networks. This approach to database development substantially reduces the effort required to acquire the information upon which neural networks may be trained.
- The basic operations, and how these combine to make predictions of mechanical responses, in an FFNN were outlined. This includes the role of weights, biases and activation functions as well as the description of the training stage of the neural network as a minimization problem using notation common within the mechanical sciences.
- A similar description of convolutional neural networks was developed for two different possible applications: (1) to solve inverse problems where the boundary conditions need to be identified from a known stress distribution and (2) as a classifier to identify if damage will occur within a microstructure given a known stress distribution.
- Two microstructure-sensitive topology optimizations are demonstrated. In the first case, the material response at the microscale is derived from the FFNN results, and used to perform design against a load that causes the material to behave in a non-linear elastic way. In the second case, a material damage constrain is added to the optimization, where the CNN is used to identify if damage has occur on the microscale and penalize the design accordingly.

In short, we have provided methods to more rapidly produce the data needed to train neural networks, developed further insight into the working of neural networks from a mechanical sciences perspective, and highlighted the potential for these methods to enhance practical design tasks. The database of responses made with SCA, code used for training and prediction with the neural networks, and the topology optimization codes are available at https://github.com/wing-kam-liu-group. We hope that this will encourage the use of data science and machine learning as a tool for mechanistic analysis, rather than simple as an unknown black-box operator.

5.2 Future work

Several areas where further investigation might be useful have already been noted:

- Further development of clustering methods to represent large deformations, better capture anisotropic behavior or behavior that changes due to loading conditions, and even refine clusters during the prediction stage might be promising. The development of contact or self-contact formulations applicable to clustering discretization methods would aid in generality.
- Formulations of clustering discretization solutions applicable to the component scale (rather that only the RVE scale), and the extension of concurrent multiscale solutions that use clustering discretization at multiple scales are currently under development.
- For neural networks, methods to include history-dependence (e.g. plasticity) are currently an active area. Including physics in the neural network directly is another developing area, e.g. with physics-informed neural networks (PINNs).
- For optimization, sensitivity analysis for topology optimization with FFNN and CNN should be further developed. More flexible software to support this would also be desirable.
- Multiscale topology optimization with various material microstructure databases is still a developing area. The approach outlined here may be a promising method to simultaneously optimize topology and microstructure given sufficient constraints.
- The "data-driven" component of these methods (both clustering-based discretization and neural networks) are not restricted to the use of computational data. Information from other sources, e.g. experimental sensor data and images, could be included if it is available. If mixed data

streams are used extra care in data representation would be required.
– Continuous validation and verification studies will help make these methods robust and reliable.

Acknowledgements The authors thank Sourav Saha and Satyajit Mojumder for their helpful suggestions during the writing process. W.K.L. acknowledges the support of the National Science Foundation under Grant No. MOMS/CMMI-1762035. O.L.K. thanks the United States National Science Foundation (NSF) for their support through the NSF Graduate Research Fellowship Program under financial award number DGE-1324585. H. L. acknowledges the support by the Northwestern University Data Science Initiative (DSI) under Grant No. 171 4745002 10043324. L.Z. and SQ.T. were supported partially by the National Science Foundation of China under Grant numbers 11832001, 11521202, and 11890681.

References

1. Hashin Z, Shtrikman S (1963) A variational approach to the theory of the elastic behaviour of multiphase materials. J Mech Phys Solids 11(2):127
2. Hill R (1965) A self-consistent mechanics of composite materials. J Mech Phys Solids 13(4):213
3. Ghosh S, Lee K, Moorthy S (1996) Two scale analysis of heterogeneous elastic-plastic materials with asymptotic homogenization and Voronoi cell finite element model. Comput Methods Appl Mech Eng 132(1–2):63
4. Paley M, Aboudi J (1992) Micromechanical analysis of composites by the generalized cells model. Mech Mater 14(2):127
5. Dvorak GJ (1992) Transformation field analysis of inelastic composite materials. Proc R Soci London Series A Math Phys Sci 437(1900):311–327
6. Michel JC, Suquet P (2003) Nonuniform transformation field analysis. Int J Solids Struct 40(25):6937
7. Yvonnet J, He QC (2007) The reduced model multiscale method (R3M) for the non-linear homogenization of hyperelastic media at finite strains. J Comput Phys 223(1):341
8. Berkooz G, Holmes P, Lumley JL (1993) The proper orthogonal decomposition in the analysis of turbulent flows. Annu Rev Fluid Mech 25(1):539
9. Liu Z, Bessa M, Liu W (2016) Self-consistent clustering analysis: an efficient multi-scale scheme for inelastic heterogeneous materials. Comput Methods Appl Mech Eng 306:319
10. Shakoor M, Kafka OL, Yu C, Liu WK (2018) Data science for finite strain mechanical science of ductile materials. Comput Mech 1–13
11. Kafka OL, Yu C, Shakoor M, Liu Z, Wagner GJ, Liu WK (2018) Data-driven mechanistic modeling of influence of microstructure on high-cycle fatigue life of nickel titanium. JOM 70(7):1154
12. Yu C, Kafka OL, Liu WK (2019) Self-consistent clustering analysis for multiscale modeling at finite strains. Comput Methods Appl Mech Eng 349:339
13. Oishi A, Yagawa G (2017) Computational mechanics enhanced by deep learning. Comput Methods Appl Mech Eng 327:327
14. Kirchdoerfer T, Ortiz M (2016) Data-driven computational mechanics. Comput Methods Appl Mech Eng 304:81
15. Liu Z, Wu C, Koishi M (2018) A deep material network for multiscale topology learning and accelerated nonlinear modeling of heterogeneous materials. Comput Methods Appl Mech Eng 345:1138–1168
16. Tang S, Zhang L, Liu WK (2018) From virtual clustering analysis to self-consistent clustering analysis: a mathematical study. Comput Mech 1–18
17. Cheng G, Li X, Nie Y, Li H (2019) FEM-Cluster based reduction method for efficient numerical prediction of effective properties of heterogeneous material in nonlinear range. Comput Methods Appl Mech Eng 348:157–184
18. Kröner E (1972) Statistical continuum mechanics, vol 92. Springer, Berlin
19. Gan Z, Li H, Wolff SJ, Bennett JL, Hyatt G, Wagner GJ, Cao J, Liu WK (2019) Data-driven microstructure and microhardness design in additive manufacturing using self-organizing map. Engineering (**in press**)
20. Zhang L, Tang S, Yu C, Zhu X, Liu W K (2019) Fast calculation of interaction tensors in clustering-based homogenization. Comput Mech (**in press**)
21. Nie Y, Cheng G, Li X, Xu L, Li K (2019) Principle of cluster minimum complementary energy of FEM-cluster-based reduced order method: fast updating the interaction matrix and predicting effective nonlinear properties of heterogeneous material. Comput Mech. https://doi.org/10.1007/s00466-019-01710-6
22. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press. http://www.deeplearningbook.org. Accessed 28 Apr 2019
23. Haykin S (1994) Neural networks: a comprehensive foundation. Prentice Hall PTR, Upper Saddle River
24. Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85
25. Funahashi KI (1989) On the approximate realization of continuous mappings by neural networks. Neural Netw 2(3):183
26. Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Netw 2(5):359
27. Ghaboussi J, Garrett J Jr, Wu X (1991) Knowledge-based modeling of material behavior with neural networks. J Eng Mech 117(1):132
28. Furukawa T, Yagawa G (1998) Implicit constitutive modelling for viscoplasticity using neural networks. Int J Numer Methods Eng 43(2):195
29. Qingbin L, Zhong J, Mabao L, Shichun W (1996) Acquiring the constitutive relationship for a thermal viscoplastic material using an artificial neural network. J Mater Process Technol 62(1–3):206
30. Yeh IC (1998) Modeling of strength of high-performance concrete using artificial neural networks. Cement Concrete Res 28(12):1797
31. Huber N, Tsakmakis C (2001) A neural network tool for identifying the material parameters of a finite deformation viscoplasticity model with static recovery. Comput Methods Appl Mech Eng 191(3–5):353
32. Pichler B, Lackner R, Mang HA (2003) Back analysis of model parameters in geotechnical engineering by means of soft computing. Int J Numer Methods Eng 57(14):1943
33. Kucerová A, Leps M, Zeman J (2009) Back analysis of microplane model parameters using soft computing methods. arXiv preprint arXiv:0902.1690
34. Feyel F, Chaboche JL (2000) FE2 multiscale approach for modelling the elastoviscoplastic behaviour of long fibre SiC/Ti composite materials. Comput Methods Appl Mech Eng 183(3–4):309
35. Kouznetsova V, Geers MG, Brekelmans WM (2002) Multi-scale constitutive modelling of heterogeneous materials with a gradient-enhanced computational homogenization scheme. Int J Numer Methods Eng 54(8):1235
36. Feyel F (2003) A multilevel finite element method (FE2) to describe the response of highly non-linear structures using generalized continua. Comput Methods Appl Mech Eng 192(28—-30):3233
37. Ghaboussi J, Pecknold DA, Zhang M, Haj-Ali RM (1998) Autoprogressive training of neural network constitutive models. Int J Numer Methods Eng 42(1):105
38. Shin H, Pande G (2000) On self-learning finite element codes based on monitored response of structures. Comput Geotech 27(3):161

39. Shin H, Pande G (2003) Identification of elastic constants for orthotropic materials from a structural test. Comput Geotech 30(7):571
40. Gawin D, Lefik M, Schrefler B (2001) ANN approach to sorption hysteresis within a coupled hygro-thermo-mechanical FE analysis. Int J Numer Methods Eng 50(2):299
41. Lefik M, Schrefler B (2002) One-dimensional model of cable-in-conduit superconductors under cyclic loading using artificial neural networks. Fusion Eng Des 60(2):105
42. Lefik M, Schrefler B (2003) Artificial neural network as an incremental non-linear constitutive model for a finite element code. Comput Methods Appl Mech Eng 192(28–30):3265
43. Le B, Yvonnet J, He QC (2015) Computational homogenization of nonlinear elastic materials using neural networks. Int J Numer Methods Eng 104(12):1061
44. Bhattacharjee S, Matouš K (2016) A nonlinear manifold-based reduced order model for multiscale analysis of heterogeneous hyperelastic materials. J Comput Phys 313:635
45. Cecen A, Dai H, Yabansu YC, Kalidindi SR, Song L (2018) Material structure-property linkages using three-dimensional convolutional neural networks. Acta Mater 146:76
46. Wang K, Sun W (2019) Meta-modeling game for deriving theory-consistent, microstructure-based traction-separation laws via deep reinforcement learning. Comput Methods Appl Mech Eng 346:216
47. Han J, Moraga C (1995) The influence of the sigmoid function parameters on the speed of backpropagation learning. In: International workshop on artificial neural networks. Springer, Berlin, pp 195–201
48. Matlab deep learning toolbox (2018b) MATLAB deep learning toolbox. The MathWorks, Natick
49. Levenberg K (1944) A method for the solution of certain non-linear problems in least squares. Q Appl Math 2(2):164
50. Marquardt DW (1963) An algorithm for least-squares estimation of nonlinear parameters. J Soc Ind Appl Math 11(2):431
51. Goodfellow I, Bengio Y, Courville A (2017) Deep learning. MIT Press, Cambridge
52. Matsugu M, Mori K, Mitari Y, Kaneda Y (2003) Subject independent facial expression recognition with robust face detection using a convolutional neural network. Neural Netw 16(5–6):555
53. Lubbers N, Lookman T, Barros K (2017) Inferring low-dimensional microstructure representations using convolutional neural networks. Phys Rev E 96(5):052111
54. Kondo R, Yamakawa S, Masuoka Y, Tajima S, Asahi R (2017) Microstructure recognition using convolutional neural networks for prediction of ionic conductivity in ceramics. Acta Mater 141:29
55. DeCost BL, Francis T, Holm EA (2017) Exploring the microstructure manifold: image texture representations applied to ultrahigh carbon steel microstructures. Acta Mater 133:30
56. Ling J, Hutchinson M, Antono E, DeCost B, Holm EA, Meredig B (2017) Building data-driven models with microstructural images: generalization and interpretability. Mater Discov 10:19
57. Cang R, Li H, Yao H, Jiao Y, Ren Y (2018) Improving direct physical properties prediction of heterogeneous materials from imaging data via convolutional neural network and a morphology-aware generative model. Comput Mater Sci 150:212
58. Yang Z, Yabansu YC, Al-Bahrani R, Wk Liao, Choudhary AN, Kalidindi SR, Agrawal A (2018) Deep learning approaches for mining structure-property linkages in high contrast composites from simulation datasets. Comput Mater Sci 151:278
59. Yang Z, Yabansu YC, Jha D, Wk Liao, Choudhary AN, Kalidindi SR, Agrawal A (2019) Establishing structure-property localization linkages for elastic deformation of three-dimensional high contrast composites using deep learning approaches. Acta Mater 166:335
60. Abaqus V (2014) 6.14 Documentation. Dassault Systemes Simulia Corporation 651
61. Cheng G, Guo X (1997) ε-relaxed approach in structural topology optimization. Struct Optim 13(4):258
62. Duysinx P, Bendsøe MP (1998) Topology optimization of continuum structures with local stress constraints. Int J Numer Methods Eng 43(8):1453
63. Guo X, Cheng G, Yamazaki K (2001) A new approach for the solution of singular optima in truss topology optimization with stress and local buckling constraints. Struct Multidiscip Optim 22(5):364
64. Guo X, Zhang WS, Wang MY, Wei P (2011) Stress-related topology optimization via level set approach. Comput Methods Appl Mech Eng 200(47–48):3439
65. Sigmund O (2001) A 99 line topology optimization code written in MATLAB. Struct Multidiscip Optim 21(2):120