npj computational materials

www.nature.com/npjcompumats

ARTICLE OPEN

Deep material network via a quilting strategy: visualization for explainability and recursive training for improved accuracy

Dongil Shin[1], Ryan Alberdi[2], Ricardo A. Lebensohn[3] and Rémi Dingreville[1]✉

Recent developments integrating micromechanics and neural networks offer promising paths for rapid predictions of the response of heterogeneous materials with similar accuracy as direct numerical simulations. The deep material network is one such approaches, featuring a multi-layer network and micromechanics building blocks trained on anisotropic linear elastic properties. Once trained, the network acts as a reduced-order model, which can extrapolate the material's behavior to more general constitutive laws, including nonlinear behaviors, without the need to be retrained. However, current training methods initialize network parameters randomly, incurring inevitable training and calibration errors. Here, we introduce a way to visualize the network parameters as an analogous unit cell and use this visualization to "quilt" patches of shallower networks to initialize deeper networks for a recursive training strategy. The result is an improvement in the accuracy and calibration performance of the network and an intuitive visual representation of the network for better explainability.

npj Computational Materials (2023)9:128 ; https://doi.org/10.1038/s41524-023-01085-6

INTRODUCTION

Data-driven approaches and advances in machine-learning algorithms are emerging techniques sought out to speed up the computational modeling and time-to-solution predictions of microstructure and materials behavior[1–7]. In computational mechanics, these techniques bypass computationally expensive direct numerical simulation (DNS) solvers, such as the finite element method[8], fast Fourier transform (FFT)[9–11], or mesh-free solvers[12], by approximating the effective constitutive response of materials microstructures with surrogate models trained on stress-strain datasets. For example, recent studies have put forward such surrogate models capable of discovering unknown constitutive laws[13] or rapidly predicting the effective nonlinear material's behavior[14] as well as path-dependent behavior[15–17] of microstructures. These surrogate models are based on a variety of methods, including artificial neural networks[18–20], two-dimensional (2D) and three-dimensional (3D) image-based convolutional neural networks[21,22], cluster-based reduced-order models[23–25], and Gaussian process regression models[26,27]. Alternatively, other efforts focus on accelerating DNS simulations bypassing DNS with machine-learning solvers[4,6,7,19,28,29]. However, all of these methods rely heavily on computationally expensive microstructure-level simulations for a given and fixed material constitutive relation. In other words, these machine-learning models need to be retrained whenever the constitutive relation changes, limiting their extrapolation performance.

Compared to these methods, recent works on the so-called deep material network[30–34] (DMN) circumvent this reliability on retraining the network to predict the nonlinear and inelastic responses of microstructures by using a microstructure-aware binary tree-type network with connected mechanistic building blocks (Fig. 1a). These building blocks use analytical homogenization solutions to describe the overall material response of a fixed microstructure. In the training mode, also referred to as the offline training mode in machine-learning parlance, the network is trained on linear elastic data for a given microstructure that are generated by DNS (top two panels of Fig. 1a). During this training procedure, the DMN learns the homogenization pathway and mechanical local interactions in that given microstructure, along with the microstructure-informed/physics-informed network structure. Once trained, the network does not need to be retrained and acts as a reduced-order model that can extrapolate the material's behavior to a variety of material constitutive laws (bottom panel of Fig. 1a). Specifically, in that configuration, the network parameters are pre-determined and fixed from the offline training mode, and the network instead iteratively solves the (nonlinear) stress-strain response given a (nonlinear) constitutive relationship defined for the base nodes. This application of the network as a reduced-order surrogate model is referred to as the online prediction mode. More complete descriptions of DMN are provided in the original paper by Liu and Koishi[30], the micromechanics perspective of DMN by ref. [33], and in the Methods section of this paper. This approach has been successfully applied to a wide range of problems, including multi-phase composites[32,34,35], woven structures[36], or porous materials[37], and expanded to predict the thermomechanical behavior of composites[38] or interfacial failure[39] by considering cohesive layers. Yet, even for this successful approach, two outstanding challenges have been to (i) improve the network calibration (i.e., reducing model uncertainty for the online prediction) without compromising predictive accuracy (i.e., model error from cost function during the offline training mode), especially if one wants to keep a network with a small number of layers and (ii) address explainability of the network (i.e., the ability to understand and interpret the network parameters and their role regarding the network's function, in the present case predicting the material's mechanical response). Reducing the calibration error is important because it improves the accuracy and reliability of the network's predictions. Explainability is important for the trustworthiness, debugging, and performance improvement of the network. On the topic of accuracy and calibration, the efficacy of random initialization of the network cannot guarantee reliable solutions with low uncertainty without a sufficiently deep network,

[1]Center for Integrated Nanotechnologies, Sandia National Laboratories, Albuquerque, NM 87185, USA. [2]Simulation Modeling Sciences, Sandia National Laboratories, Albuquerque, NM 87185, USA. [3]Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87845, USA. ✉email: rdingre@sandia.gov

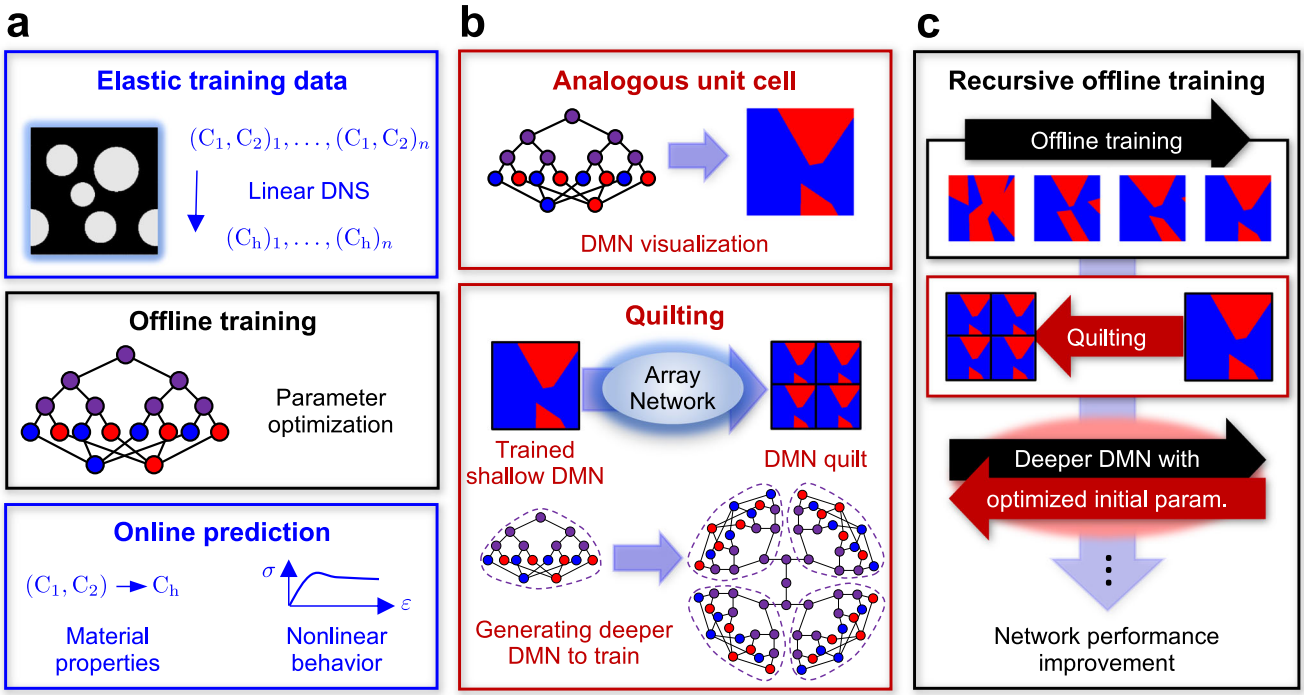

Fig. 1 Deep material network (DMN) via a quilting and recursive strategy. a A database of elastic properties is generated via direct numerical simulations (DNS). This database is used for (offline) training of the network. The trained network does not need retraining for the (online) prediction and is able to extrapolate the material's behavior to unknown materials and to various local material constitutive laws, such as plasticity or fatigue. **b** The network is visualized as an analogous unit cell constructed directly from the network parameters. Patches of these unit cells are quilted together as an array using this representation to form the initial architecture of deeper DMNs. **c** The network visualization and quilting strategy are used to recursively train increasingly deeper networks, resulting in improved performance of the network as compared to when it is randomly initialized.

substantial training optimization, and ensemble averaging of multiple trained networks[40] (i.e., combining the predictions of multiple models to produce a more accurate prediction) to strike a good balance between accuracy and calibration. Training networks via classical gradient-based optimization for back-propagation forces the network to depend on the initial network parameters[41], a fundamental limitation of randomly initializing parameters. Indeed, even though the network potentially possesses a global solution with good performance, a gradient-based optimization approach will find one of the local optima in the vicinity of the initial parameters. When the network becomes deeper, the variance between local optimum points may become smaller; however, some studies suggest a tendency for larger, more accurate models to be worse calibrated[42–44], emphasizing the importance of reducing the risk of being trapped in a bad local optimum. The explainability of the network parameters (weights and normal vectors at nodes for the DMN, see Methods for further details) is usually challenging[45] and non-intuitive, since DMN is primarily used as a "black-box" surrogate alternative to DNS solvers. The network is mathematically complex, causing difficulties in describing the meaning and values of the nodes directly and interpreting what happens during the network's parameter optimization.

Inspired by the success of recursive optimization in other scientific domains[46–48], we present a strategy for improving the accuracy and explainability of the DMN. Figure 1 illustrates the various elements of our approach. We call this approach recursive training via a quilting strategy since the network architecture is trained by "quilting" an array of unit-cell analogs patches (i.e., the visualization of previously trained shallower networks represented as analogous unit cells) to initialize the architectures of deeper networks with more layers and recursively train deeper and deeper networks. We visualize the network by constructing an analogous unit cell based on the physical interpretation of the network parameters (activation weights and normal vectors) at each layer: the activation weights are used for selecting the phase fraction of material domains, and the normal vectors are used for defining the associated orientations of those domains (Fig. 1b). Based on this visualization, we quilt together shallow optimized networks as patches of analogous unit cells representing the network as the base building blocks for constructing and initializing deeper networks (Fig. 1b, c). This process is repeated to recursively train increasingly deeper networks with optimized initial parameters. This recursive training process is applicable to both 2D and 3D DMNs. We show that this recursive quilting strategy outperforms the results obtained by randomizing the initial network parameters. The performance results from the network trained recursively illustrate that not only the network acts as a high-fidelity reduced-order model of the macroscopic, homogenized response of the original periodic cell, but that it also improves the predictions of the local behavior in terms of distribution of local stresses and strains as compared to a network trained via random initialization. We, therefore, obtain improved accuracy and calibration performance for a diverse set of 2D and 3D DNS periodic unit cells, demonstrating the applicability and generalization of this strategy. The visualization of the network as a unit cell not only serves as the key ingredient for the purpose of recursive training but also enables us to comprehend the network behavior more intuitively. Indeed, we show that this visualization can be used to track the training progress by visually providing additional insights on the convergence of the training procedure. Additionally, by comparing the DNS results on an original periodic unit cell with the DNS results performed on the network unit-cell analog, we show that this visualization not only serves as an intuitive method to explain the network, but it also correctly captures how the network constructs the interactions between

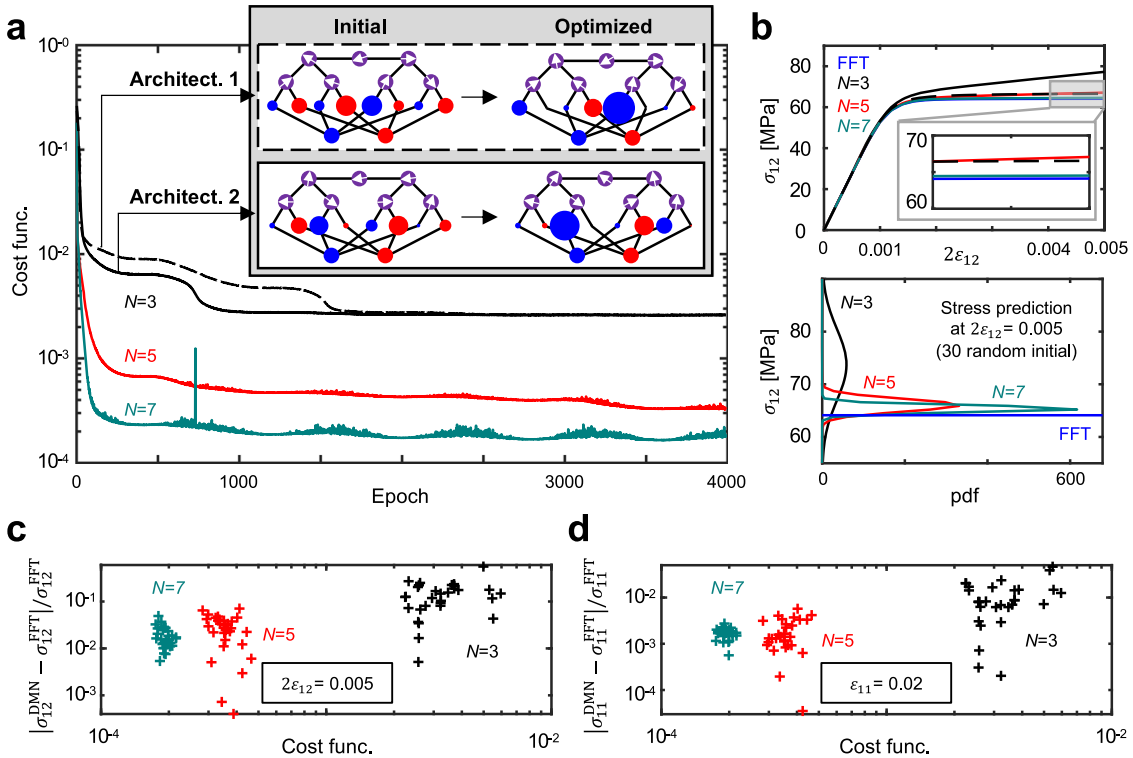

Fig. 2 Deep material network performance when randomly initialized. These results correspond to the DNS periodic unit cell presented in panel **a** of Fig. 1. **a** Training errors as a function of the number of epochs for networks with various depths ($N = 3$, 5, and 7). Deeper networks have improved accuracy. Inset in this panel illustrates two 3-layer-deep DMNs with different initial parameters (left). The resulting optimized parameters are displayed on the right. The size of the nodes scales with the values of the weights, and white arrows show the orientations of the normal vectors. **b** Online prediction results for the Norton viscoplastic constitutive relationship under shear loading conditions for the DMN trained in **a** (top panel). The distribution of the stresses at a shear strain of $2\varepsilon_{12} = 0.005$ (obtained by randomly initializing 30 different networks for $N = 3$, 5, and 7) illustrates the calibration error (bottom panel). Deeper DMN predicts the stress behavior more precisely with lower uncertainty. These results illustrate the calibration performance of the extrapolated predictions in the nonlinear regime. Accuracy (x-axis) *vs.* calibration (y-axis) error in the case of shear loading (**c**) and tensile loading (**d**) for networks with various depths. Cross symbols indicate performance for randomly initialized networks.

material phases, to the point where, under certain circumstances, it can reproduce both the macroscopic, homogenized response and the statistical local behavior of the original periodic unit cell.

RESULTS

In what follows, we start by highlighting the deficiencies of randomly initializing a DMN in terms of inconsistencies of the training accuracy and variability for the online predictions. We then illustrate how the network parameters can be visualized as an analogous unit cell by taking advantage of the network's mechanistic building blocks. We finally show how to use this visualization recursively to initialize an increasingly deeper network in order to train the network and reduce the calibration error.

For all these results, as detailed in Methods, the training data were obtained from a homogenized material property calculated by DNS FFT simulations on periodic unit cells for which each of the material phases were considered to be orthotropic. For the online predictions, in which the network acts as a reduced-order model, we used an elasto-viscoplastic constitutive relation with visco-plastic behavior given by Norton's law[49,50]. Implementation of the network for both the offline training and online predictions are based on the DMN architecture proposed by ref. [32] and ref. [34] and also described in Methods.

Accuracy and calibration with random initialization

We first start by illustrating the performance of the DMN when it is randomly initialized (see details of the network architecture in Methods). We compare the accuracy and calibration errors of networks with different depths both in the offline and online inference modes to reveal areas of improvement in the network performance. Here, for a given network of depth N ($N = 3$, 5, or 7), we trained 30 different randomly initialized networks using the DNS periodic unit cell and protocol shown in Fig. 1a. This microstructure corresponds to a binary composite containing five spherical inclusions.

Figure 2a shows the training history for randomly initialized networks of different depths. We show the mean value of the training loss for networks with N layers averaged over 30 different randomly initialized networks. These training results primarily measure the difference between the effective elastic stiffness matrix obtained by DNS and that predicted by the network (see definition of the training cost function in Eq. (21) and details of the training procedure in Methods). We observe that, after a rapid decrease of the cost function during the early stage of training (first 200 epochs), the network performance keeps improving its accuracy until it converges after at most 2000 epochs to reach a steady-state minimum error (cost $= \sim 2.5 \times 10^{-3}$, $\sim 3.2 \times 10^{-4}$, $\sim 1.8 \times 10^{-4}$ for $N = 3$, 5, 7). Even for the shallow three-layer-deep network (i.e., $N = 3$), we achieve an error of less than one percent upon completion of the training. The inset in panel **a** shows two examples of the distribution of weights in the base layer (the size of the nodes is commensurate to the activation

weights) and orientation vectors in the other layers before and after training for two 3-layer-deep networks initialized with different random seeds. We notice that, even though these two networks have comparable training accuracies, the distribution of optimized network parameters is substantially different depending on how the network was initialized. Such difference is a direct consequence of using a gradient-based optimizer to train the network and points to the fact that the trained networks settled in separate local optima, conditioned by their initial network parameters.

Based on these training results on elastic properties data, one would expect good extrapolation performance for the online prediction when using the DMN for nonlinear analyses. We show a comparison of the online predictions for a Norton viscoplastic constitutive relation under shear loading (see description in Eq. (1) and implementation in Methods) for networks with different depths in Fig. 2b. The top panel shows the predicted average stress-strain responses and the bottom panel shows the distribution of stresses at a shear strain $2\epsilon_{12} = 0.005$ across the 30 networks, for each different depths. While the average strain-stress response compares relatively well with that of the DNS (especially for deeper networks with many layers), we note that the distribution of predicted stresses amongst the 30 different randomly initialized networks, even at a relatively small shear strain ($2\epsilon_{12} = 0.005$), varies substantially depending on the depth of the network. This assertion is especially true for the 3- and 5-layer-deep networks (i.e., $N = 3$ and $N = 5$), indicating a poor calibration of the model, while the seven-layer-deep network (i.e., $N = 7$) has a narrower distribution indicating a better calibration of the model. For the shallower networks, in some cases, the 3-layer-deep network can have better-extrapolated results than the five-layer-deep network, as indicated by the overlap of distributions, but these results again are highly dependent on how the network has been initialized. By plotting the training accuracy error (x-axis) against the calibration error (y-axis) in Fig. 2c (shear loading to $2\epsilon_{12} = 0.005$) and in Fig. 2d (tensile loading $\epsilon_{11} = 0.02$), we further illustrate the point that, while increasing the number of layers reduces both the accuracy and calibration errors, shallow networks still do present relatively high calibration errors, even in the present case for a simple constitutive relation such as the Norton viscoplastic relation and for small plastic strains. It is reasonable to assume from these results that such error would be even more pronounced for larger deformations, for more complicated nonlinear constitutive relationships, or for complex loading paths for which the calibration error would accumulate and drift.

Network visualization as an analogous unit cell

The results above illustrate the difficulty of balancing an accurate model during the training procedure with a well-calibrated model determined for online predictions. Part of this issue stems from the lack of explainability during training, if one tries to interpret the nodes of the network directly through the lens of weights and biases in order to improve the network performance. As a solution, we can exploit the DMN architecture directly as a feature for explainability since it is based on micromechanics building blocks. As described in Methods, the DMN is a binary tree-type network for which, at a given layer $i-1$, nodal weights (ω^n_{i-1}) are inherited from activated nodes in layer i. The weights in that layer are directly related to the phase fractions of materials ($f^{2n}_{i-1} = \omega^{2n}_{i-1}/(\omega^{2n}_{i-1} + \omega^{2n-1}_{i-1})$). Additionally, nodes and layers also carry normal vectors $\vec{n}^n_{i-1}$ as part of the network parameters. These two elements (ω_{i-1}, $\vec{n}_{i-1}$), defined at each layer, provide us with the means to visualize the network as a unit cell from the network perspective. We define this visualization as the *DMN unit-cell analog*. It should be noted that this analogous unit cell is not periodic in contrast to the one used in DNS.

Figure 3a illustrates the DMN visualization process for a 2-layer-deep network (i.e., $N = 2$). The output layer, **Layer 0**, represents the homogenized material with the predicted homogenized materials properties obtained through DMN. As such, a visualization of the network, as seen from this layer, can be simply represented as a single homogenized material block with weight ω^1_0 (and thereby phase fraction $f^1_0 = 1$). The next layer, **Layer 1**, contains three pieces of information: two weights, ω^1_1 and ω^2_1, and a normal vector, $\vec{n}^1_0$ from Layer 0. Note that the subscript indicates the layer number, and the superscript denotes the node number. The weights can be used to define two blocks of materials of size $f^1_1 = \omega^1_1/(\omega^1_1 + \omega^2_1)$ (pink) and $f^2_1 = \omega^2_1/(\omega^1_1 + \omega^2_1)$ (green) respectively. The normal vector can be used to define the plane's orientation separating these two blocks of materials. Finally, the base layer, **Layer 2**, can be thought as describing the two separate blocks in **Layer 1** with information ($\omega^1_2, \omega^2_2, \vec{n}^1_1$) for one of the block (green block in Fig. 3a) and information ($\omega^3_2, \omega^4_2, \vec{n}^2_1$) for the other block (pink block in Fig. 3a). Following

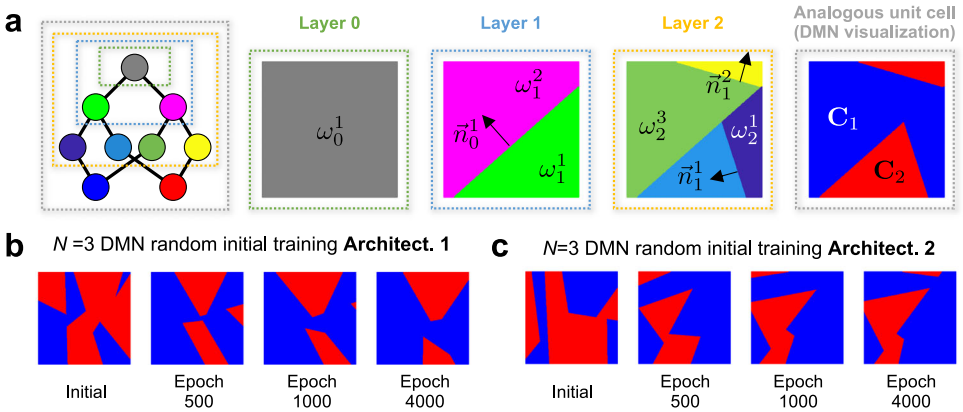

Fig. 3 Network visualization. a Visualization for a 2-layer-deep network (i.e., $N = 2$). **Layer 0** represents the homogenized material. **Layer 1** illustrates the preceding layer with two nodes with activated weight ω^1_1 and ω^2_1, and a normal vector direction $\vec{n}^0_1$. **Layer 2** splits the pink and green domains present in **Layer 1** into sub-domains based on the weights and normal vectors. By assigning material $\mathbf{C}_1$ (e.g., phase colored in red) and $\mathbf{C}_2$ (e.g., phase colored in blue) to odd and even nodes, respectively, the visual representation learned by the network becomes apparent. **b**, **c** Training history for the two 3-layer-deep architectures presented in Fig. 2a. As the number of epochs increases, the DMN unit-cell analog representation keeps evolving toward an optimized representation. The final representations are different between the two architectures depending on the initial parameterization of the DMN unit-cell analog prior to the training.

the same procedure, we can define four materials blocks based on the activation weights to select the phase fraction of each block, and the normal vectors to define the associated orientations of those blocks. Finally, by assigning materials 1 ($\mathbf{C}_1$ in blue) and 2 ($\mathbf{C}_2$ in red), respectively, to the odd and even nodes in **Layer 2**, we obtain a visualization of the network as a unit-cell analog as seen through the lens of the DMN. This visualization process can be generalized for any depth of the DMN by repeating it for each layer in the network before assigning two materials in the final layer. This approach can be applied to DMNs trained on 3D DNS periodic unit cells, updating the blocks of materials to be volume and the normal vectors to cut the volumes with a normal plane. Note that the above description of the visualization of the DMN is not unique since there is no intrinsic ordering between the phases, and as such, other visual constructs could be proposed. The analogous unit cell representation of the network is not meant to reconstruct a realization of the microstructure of the original DNS unit cell. Rather, this visual approach offers a straightforward and intuitive visualization, which is easy to follow and based on the mechanistic kinematic and kinetic relationships built into the logic of the network itself.

initial random DMN unit cell. We can easily observe that the features of the optimized DMN unit cell are rapidly found during the training history, correlating with the rapid drop in the cost function. After this initial step, as the number of epochs increases, the DMN unit-cell analog converges towards a final representation that remains close to the initial DMN unit cell. The final representations are different between the two architectures depending on the initial parameterization of the network.

Recursive training by quilting analogous DMN unit cells

The visualization process described above constitutes the primary ingredient for constructing and initializing deeper networks in our quilting strategy for recursive training. As illustrated in the bottom panel of Fig. 1b, generating deeper networks trained from shallower DMNs could be visually represented as quilting patches of DMN unit-cell analogs as an array. In other words, we consider the updated initial deeper networks as a DMN unit cell assembled from pre-optimized and pre-homogenized smaller DMN unit-cell analogs.

Algorithm 1. Recursive DMN training strategy via quilting for a 2D problem.

Input : Initial numbers of layers (N^i), Target numbers of layers (N^t)
Output : Trained DMN for depth N^t

$N = N^i$
Initialize N-layer DMN model network parameters (ex. $\text{DMN}_1 \rightarrow (\omega_1^1, \omega_1^2) = 0.5, \vec{n}_0^1 = [0,1]$)
while $N \leq N^t$ **do**
  DMN offline training: $\text{DMN}_N \rightarrow \text{DMN}_N^*$
  Save trained network parameters: $({}^*\omega_N^1, \ldots, {}^*\omega_N^{2^N}, {}^*\vec{n}_0^1, {}^*\vec{n}_1^1, \ldots, {}^*\vec{n}_{N-1}^{2^{N-1}})$
  Generate deeper DMN: $N = N + 2$
  Initialize DMN_N recursive parameters:
  for i 0 to $N-3$ **do**
    for j 1 to 2^i **do**
      $(\vec{n}_{i+2}^{0\cdot 2^i+j}, \vec{n}_{i+2}^{1\cdot 2^i+j}, \vec{n}_{i+2}^{2\cdot 2^i+j}, \vec{n}_{i+2}^{3\cdot 2^i+j}) = {}^*\vec{n}_i^j$
    end
  end
  for j 1 to 2^{N-2} **do**
    $(\omega_N^{0\cdot 2^N+j}, \omega_N^{1\cdot 2^N+j}, \omega_N^{2\cdot 2^N+j}, \omega_N^{3\cdot 2^N+j}) = {}^*\omega_{N-2}^j/4$
  end
  Initialize DMN_N array network parameters: $\vec{n}_0^1 = [0,1], \vec{n}_1^1 = \vec{n}_1^2 = [1,0]$
end

This construct allows us to visualize the neural network as a unit cell at any time during the training history, offering a deeper understanding of the convergence of the optimized network parameters. For instance, in Fig. 3b, c, we show the evolution of the DMN unit-cell analog during training for the two 3-layer-deep network architectures discussed in the previous Results section in Fig. 2. This visual comparison illustrates our previous conclusion directly and visually: the set of initial parameters is the dominant factor leading to the set of converged and optimized network parameters. Alternatively, this conclusion can be reframed from a different perspective based on our visualization process: the initial random DMN unit-cell analog is the dominant factor leading to an optimized DMN unit cell. Indeed, for both architectures, the final optimized DMN unit cell presents reminiscent features of the

As depicted in Fig. 4a, we "quilted" an array of patches of DMN unit-cell analogs obtained from previously trained and optimized shallower networks to initialize the architecture of a deeper network with more layers. The illustration in Fig. 4a shows an example of how to construct a four-layer-deep network from a two-layer-deep network. In the 2D case, the initialization process goes as follows. The previously trained DMN is replicated four times, updating each weight to be a quarter of the original weight in order to have a consistent total weight ω_0^1 (see DMN architecture in Methods). Hereafter, by adding two layers at the top of the network, we form a 2×2 array network of the previously trained DMN. We set the angle for the normal vector for the array network to be $\pi/2$ for layer 0 and zero for layer 1 (zero for layer 0 and $\pi/2$ for layer 1 is also an option). Once this deeper network has been initiated, it can be trained in a regular

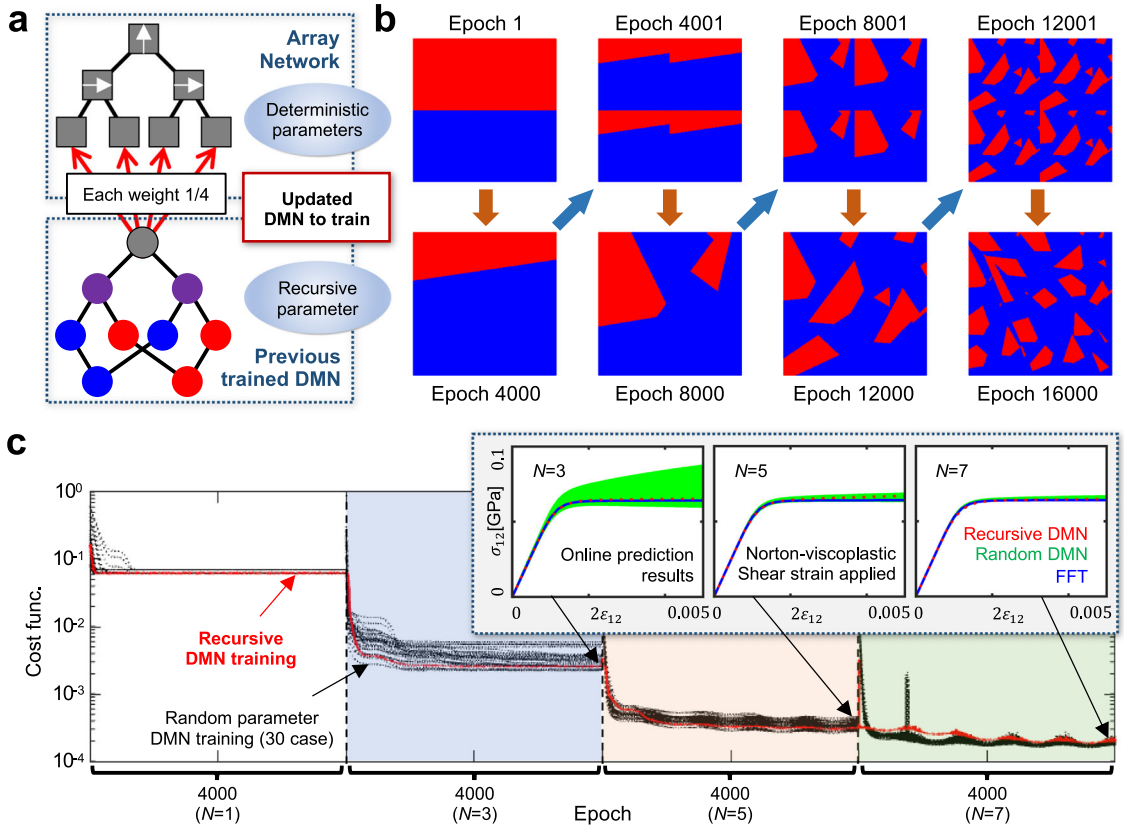

Fig. 4 Recursive training performance via a quilting strategy. a Example of DMN recursive training showing how to construct and initialize a $N = 4$ DMN from a $N = 2$ DMN in 2D-microstructure training. After training the shallower network, the optimized network is used for quilting an array network. The array network is initialized with a quarter of each weight of the shallow optimized DMN unit cells and the known normal vectors. **b** Recursively training of deeper and deeper networks ($N = 1$, 3, 5, and 7) for 4000 epochs by quilting patches of optimized DMN unit cells to initialize deeper networks at 4001, 8001, and 12,001 epochs. Orange arrows indicate the recursive training steps, and blue arrows the quilting steps. **c** Evolution of the training error achieved via recursive training compared to the error obtained from random initialization of the DMN parameters. Insets show the online predictions using Norton viscoplastic constitutive relationship for networks with different depths.

fashion in offline training as described in Methods. This process is repeated to achieve deeper and deeper networks. The pseudo-code formalizing and detailing of this process is provided in Algorithm 1.

Figure 4b illustrates this recursive training as a function of the training history via the visualization of the DMN. Here, we started from a single-layer-deep network (i.e., $N = 1$) at epoch 1, setting the weights of the only two nodes in the network to be the same with a normal vector set to $\pi/2$. After training this single-layer network for 4000 epochs, we constructed a three-layer-deep network by quilting a 2×2 array from the learned $N = 1$ DMN unit-cell analog. This sequential training/quilting process was repeated until we successfully trained and optimized our deepest network. To avoid perfect symmetry after the quilting step, we added a small perturbation (10^{-6}) on top of the normal vector values before starting the offline training. A video of the entire recursive training sequence is provided in the Supplementary Information. A deeper DMN has an additional number of network parameters which increase the network's ability to fit the training data. However, at the same time, additional degrees of freedom in the base layer increase the computational cost of the model, and has, therefore, more chance to fall into local solutions. One of the advantages of this training strategy is that the initial optimization of a network of depth N starts from a locally optimized set of network parameters from shallower networks. By recursively increasing the number of layers via quilting, we gradually expand the solution space for parameter optimization from already well-defined local optima, reducing the chances for the network to be trapped in a local optima and *de facto* improving the quality of training and associated accuracy of the model.

This last point is demonstrated in Fig. 4c by comparing the training history between the recursive training approach and that when the network is instead randomly initialized. For a given network of depth N ($N = 1$, 3, 5, and 7), we trained and optimized the N-layer-deep network for 4000 epochs, resulting in a total of 16 000 epochs to train a seven-layer-deep network via our recursive approach. The solid red line indicates the training error for the recursive approach, and the thin black lines show the training errors for 30 different randomly initialized networks following the same procedure as in Fig. 2. We observe that the recursive training converges more slowly than the random initialization training, but this slower convergence is insignificant. Despite this slower convergence, the recursive training consistently outperforms most training errors when the network is randomly initialized. In other words, the recursive training strategy consistently finds a better local solution to optimize the network parameters amongst the wide range of multiple local solutions obtained by randomly initializing the network. This improvement is especially noticeable for shallow networks. Indeed, when $N = 3$, we note a lot of spread in the training error for the networks which have been randomly initialized. Also, while the results with random initialization training for $N = 5$ and 7 converge to similar training errors, the improved training error obtained via recursive training indicates that this strategy can achieve errors around the mean value of the random initialization training results. Such performance highlights that the recursive approach enables us to filter out and avoid bad local training results.

Table 1. Comparison of local stress distributions.

	Matrix - mean [MPa]	Matrix - std. [MPa]	Inclusion - mean [MPa]	Inclusion - std. [MPa]	Vol. frac. [%]	Macro [MPa]
DNS (FFT)	56.69	7.80	82.32	15.90	0.29	64.18
$N = 3$ Random	50.97 ± 7.65	13.98 ± 10.65	179.54 ± 70.99	183.17 ± 172.83	0.29 ± 0.00	88.48 ± 19.38
$N = 5$ Random	55.42 ± 1.30	10.28 ± 2.14	96.98 ± 10.93	64.83 ± 58.21	0.29 ± 0.00	67.57 ± 2.77
$N = 7$ Random	54.70 ± 0.63	11.08 ± 0.93	93.37 ± 4.07	48.13 ± 17.55	0.29 ± 0.00	66.00 ± 0.91
$N = 3$ Recursive	58.71	1.84	81.24	40.60	0.29	65.29
$N = 5$ Recursive	58.47	4.65	87.93	57.24	0.29	67.06
$N = 7$ Recursive	58.58	4.57	79.90	31.50	0.29	64.83

DNS results show the mean and standard deviation of the stress local distribution for the matrix and inclusion, respectively. Distribution of local stresses obtained directly from the nodes of the networks are listed for DMN randomly initialized (Random) or via the recursive training strategy (Recursive). In the case of randomly initialized networks, results are averaged over 30 different networks which were initialized differently.

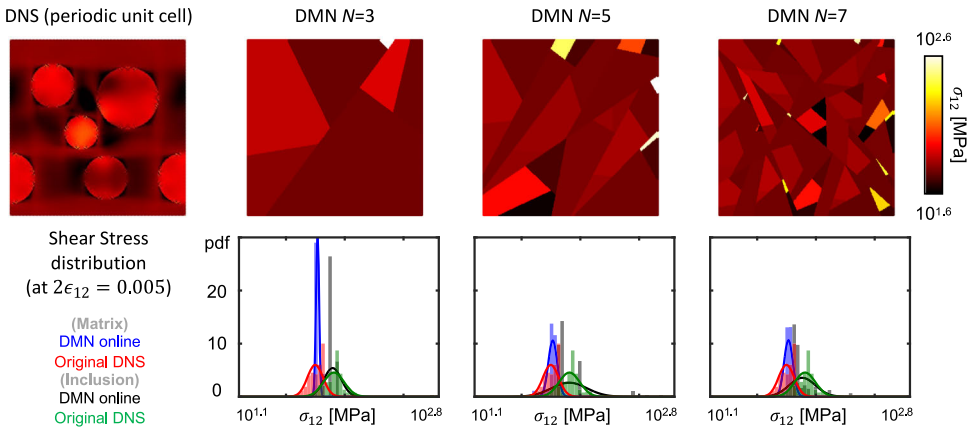

Fig. 5 Local stress distribution comparison. Illustration of the local stress σ_{12} within the original periodic unit cell and within the DMN as visualized using the analogous unit cell when $2\epsilon_{12} = 0.005$. The visualization of the network as unit cells corresponds to the recursive training results discussed in Fig. 4. DMN results represent the shear stress corresponding to each base node in the network. A quantitative comparison can be found in Table 1. Histograms represent the distributions of local stresses in the matrix and inclusion within the network and are compared to the same distributions from the DNS results on the original periodic unit cell.

We also looked at the calibration performance of the network for online prediction. The insets in Fig. 4c show comparisons of the predictions of the plastic response for the DNS periodic unit cell shown in Fig. 1 under shear loading when the network is trained recursively (red line) or via random initialization (green). Results for the random initialization are presented as the 95th-percentile confidence over the 30 trained networks. The prediction from the DNS is provided as a reference stress-strain curve (blue line). We make two observations. First, we note that deeper networks have better agreement with the DNS results. Indeed, by increasing the number of interactions between the nodes in deeper networks, regardless of how the network is initialized, our results show that we can achieve a smaller calibration error. This is visually illustrated by the increasing number of inclusions in the optimized analogous DMN unit cell via recursive training shown in Fig. 4b. Second, we also observe that, while the uncertainty (and therefore calibration error of the network) decreases with an increasing number of layers, our recursive training DMN strategy provides outperforming predictions compared to the results via random initialization, regardless of the depth of the network. This is a noteworthy improvement in the calibration of the network, especially for the shallow network.

Finally, by comparing the local stress distribution as predicted by the DNS simulations with those extracted from nodes of the network, we emphasize the advantage of the recursive training. We tabulated this comparison in Table 1. We separated the predicted distribution of local stresses for the matrix and the inclusion, respectively. Distributions obtained from the recursive training show better agreement with the DNS ground-truth results as compared to those obtained when the network was randomly initialized. This assertion is true not only for the predicted mean value, but also for the standard deviation and for both the matrix and inclusion phases. When we validate all the DMN online predictions, selecting the best random initial parameter DMN could potentially perform better than that trained via our recursive training strategy. However, this feat obviously is not guaranteed, as it requires multiple trial and error attempts to identify such a potentially best-performing network. This also proves to be a challenging task for validating that it is indeed the best-performing network in the absence of a reference, highlighting the ability of recursive training to reduce the risk of selecting poorly trained networks.

Figure 5 visually illustrates the distribution of local shear stresses presented in Table 1 for the original periodic cell and the corresponding distribution at each base node of the network represented as an analogous unit cell obtained by recursive training. While this spatial distribution of stresses in the DMN visualization is not meant to reproduce the spatial distribution in the original DNS unit cell, it serves to show how the stresses are distributed throughout the network via a reduced number of degrees of freedom and extend our understanding of the network performance (explainability). However, as shown in Table 1 and in the histograms below the DMN visualizations in Fig. 5, while not spatially arranged the same way, distributions of the local stresses

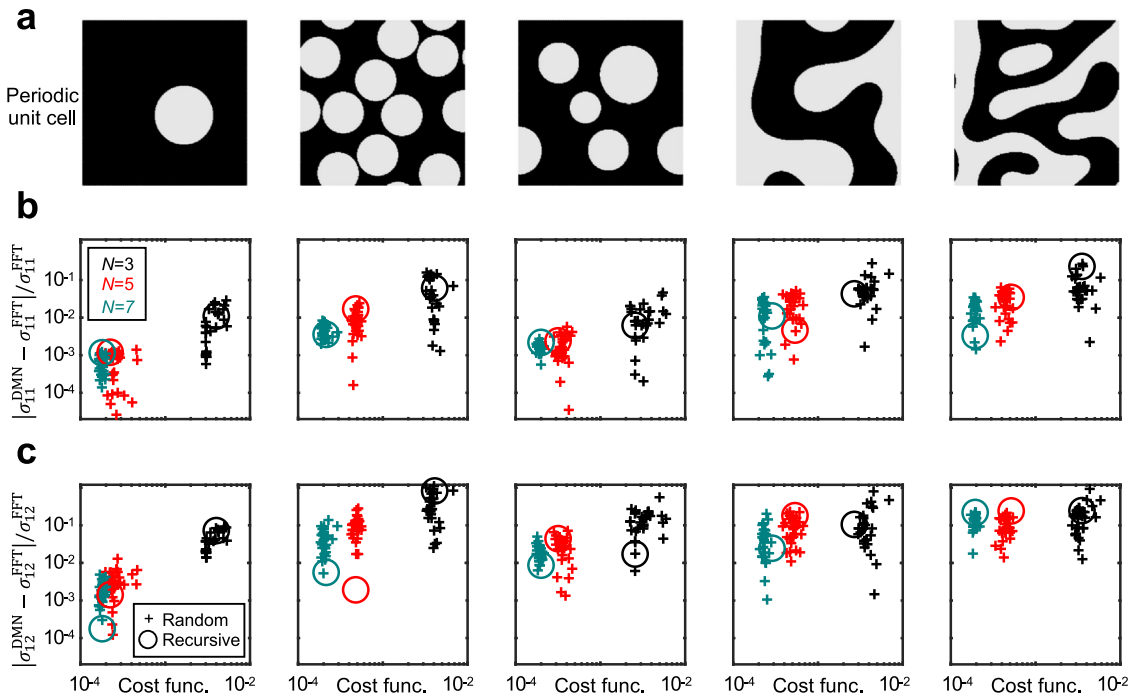

Fig. 6 Performance of the recursive DMN training on different types of 2D periodic unit cells. a Five periodic unit cells topologies with various levels of complexity. The first three periodic unit cells correspond to microstructures composed of a matrix with inclusions. The last two periodic unit cells correspond to spinodal decomposition microstructures. **b**, **c** Accuracy (x-axis) vs. calibration (y-axis) in the case of tensile loading ($\epsilon_{11} = 0.02$) and shear loading ($2\epsilon_{12} = 0.005$) for the five periodic unit cells represented above. Cross symbols indicate performance for randomly initialized networks, and the circle marker indicates performance for networks trained via our recursive training and quilting strategy.

from the original DNS unit cell and from the DMN visualization appear to be statistically similar. This is a useful feature since one could directly derive the macroscopic stress from this statistical distribution of local stresses by considering the mean local stress value of each phase weighted by the volume fraction at each node in the spirit of the homogenization process described in Eq. (9) (note that the DMN results and DNS results have the same volume fraction of phases).

Generalization of recursive training for different microstructures

We now turn to extensions and generalizations of the approach described above for different periodic unit cells problems, which have more complicated microstructural topologies and can potentially prove to be harder to train. These different DNS periodic unit cells are presented in Fig. 6a: the first three periodic units cells from the left correspond to binary microstructures composed of a matrix with inclusion, while the other two microstructures correspond to binary mixtures of co-existing phases obtained by spinodal decomposition. For the periodic units cells composed of a matrix with inclusions, the first unit cell contains one inclusion with a 10% volume fraction, the second unit cell contains ten inclusions of the same size with a 50% volume fraction, and the last unit cell (same with Fig. 1a) contains five inclusions of different sizes with 29% of volume fraction. This last configuration actually corresponds to a cross-section of the 3D periodic units cell with spherical inclusions presented later. The spinodal microstructures were generated using the phase-field method[51,52], with a 50% volume fraction for each phase. In Fig. 6, we present results for the accuracy vs. calibration error for those five 2D unit cells for two different types of loading configurations (tensile, $\epsilon_{11} = 0.02$ in Fig. 6b and shear, $2\epsilon_{12} = 0.005$ in Fig. 6c). As done in the previous section, we compare the results between random initialization training and recursive training. First, we note that the recursive training strategy consistently yields better accuracy (lower cost function) than that obtained with random initialization for most periodic unit cells (note that the axes are in log scale). This tendency is more prevalent for shallow networks, when the offline training performance accuracy is more spread for networks randomly initialized. In the case of the composite microstructures with spherical inclusions, we note that, as the networks get deeper, the recursive training strategy yields better trade-offs between accuracy and calibration errors than that obtained with random initialization, as indicated by the spread of calibration errors from random initialization even for deep networks. However, for the spinodal decomposition microstructures, the difference in performance between the recursive training strategy and random initialization is marginal, while the calibration error converges slower. These results point to an interesting observation. For topologically complex microstructures, as in the case of the spinodal microstructures, these results suggest that the homogenization pathway learned by the network does not fully capture the topological interactions emerging from the microstructure, even though the accuracy during the offline training is similar to composite microstructures with spherical inclusions. In this case, a solution to improve the network calibration without compromising its predictive accuracy is most likely to construct deeper networks (i.e., $N > 7$), with more degrees of freedom to describe the microstructural topology.

We also obtained improved performance in the case of a 3D microstructure, as shown in Fig. 7. The 3D DNS microstructure is composed of a matrix with four spherical inclusions with 20% of volume fraction, as shown in Fig. 7a inset. In this case, however, the construct for the array network needed to be updated from a 2×2 array in 2D to a $2 \times 2 \times 2$ cube in 3D to account for the additional dimensional in the DNS periodic unit cell. The previously trained network is quilted via eight DMN unit cells, updating each activated weight as 1/8 of the original weights. Additionally, we set the angle for the normal vectors for the initial

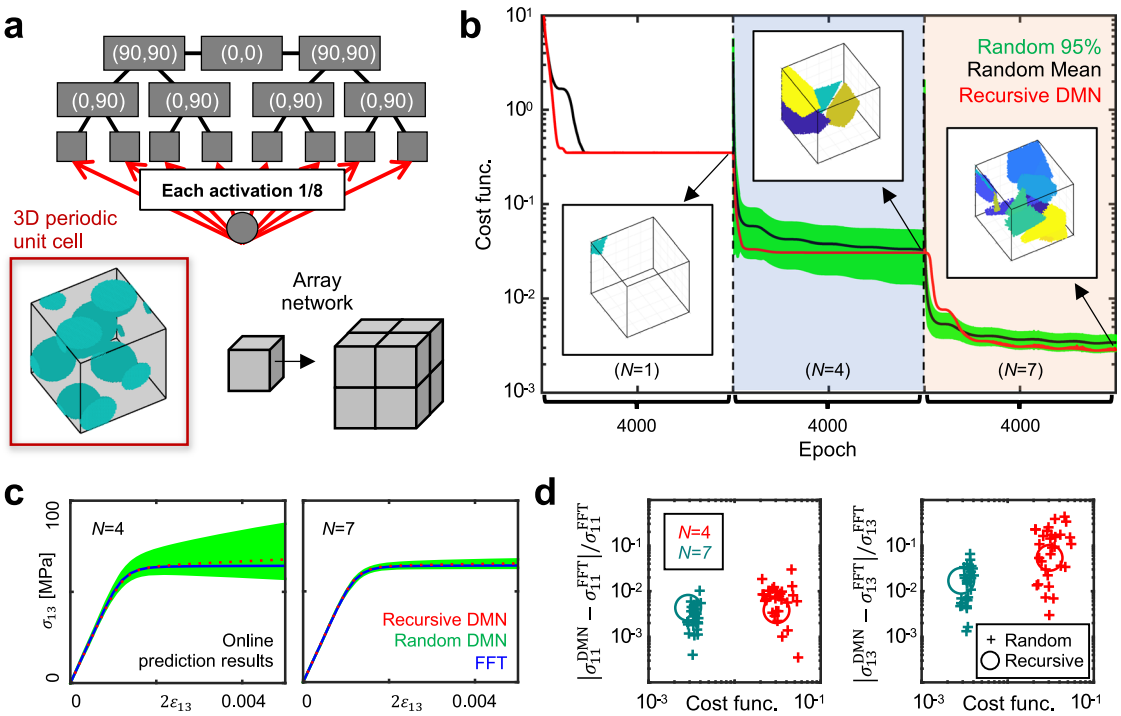

Fig. 7 Performance and visualization for 3D periodic unit cells. a Illustration of the array network for the 3D DMN along with the base 3D periodic unit cell used for training. **b** Comparison of the training history of the recursive DMN with that of random parameter initialization. Insets illustrate the optimized DMN unit cell analogs optimized for $N = 1$, 4, and 7. **c** Comparison of the online predictions between a 4- and 7-layer-deep network for a 3D microstructure subjected to a shear loading ($2\epsilon_{13} = 0.005$). **d** Accuracy (x-axis) vs. calibration (y-axis) error for 4- and 7-layer-deep network for a 3D periodic unit cell subjected to a tensile (left panel) and shear (right panel) loading. Cross symbols indicate performance for randomly initialized networks, and circle symbol indicates performance for networks trained via quilting strategy.

array cube to be (0,0) in Layer 0, ($\pi/2$, $\pi/2$) in Layer 1, and (0, $\pi/2$) in Layer 2 (note that angles are defined in spherical coordinates). As a result, each recursive iteration increases the depth of the network with three additional layers. This process is illustrated in Fig. 7a. The resulting training history (red line) is shown in Fig. 7b and compared with that of randomly initializing 30 different networks (green shade representing the 95th-percentile confidence range). We observe an even better performance than in 2D from the recursive training as compared to the random initialization strategy, including for deeper networks. This performance improvement is expected since there are a lot more network parameters to optimize in 3D. In this context, the random initialization strategy is even more subject to being trapped in a bad local optimum. In contrast, the recursive training strategy consistently finds good local optima. Additionally, the visualization of the 3D DMN unit-cell analogs (see inset in Fig. 7b, colors are used to indicate the inclusions generated from separate nodes) reveals the level of topological complexity learned by the network as a function of the number of layers. A video of the 3D recursive training sequence is provided in the Supplementary Information. Finally, results in the online prediction demonstrate the difficulty of achieving a well-calibrated model when using random initialization, an issue not as pronounced when using recursive training. In Fig. 7c, we contrast the online predictions between a 4- and 7-layer-deep network for a microstructure subjected to a shear loading ($2\epsilon_{13} = 0.005$). Clearly, the shallower network ($N = 4$) shows a lot of variability in the online prediction when randomly initialized, while the network trained through recursive training shows more robust predictions. This point is further clarified by plotting the accuracy error *vs*. the calibration error in Fig. 7d for tensile ($\epsilon_{11} = 0.02$) and shear loadings ($2\epsilon_{13} = 0.005$) on the 3D periodic unit cell. We observe that, regardless of the loading condition, the recursive training offers a good balance between accuracy (x-axis) and calibration (y-axis). For both the shallow ($N = 4$) and deep ($N = 7$) networks, the recursive training strategy with quilting initialization robustly yields smaller accuracy and calibration errors. This is not necessarily the case for networks randomly initialized since the 'bad' performing networks can yield up to 40% calibration error for the shear loading case.

DISCUSSION

We have demonstrated above that employing a recursive training strategy based on the visualization and interpretation of the network as a unit-cell analog is a robust approach to improve the accuracy and calibration of the DMN both in the offline training and online prediction modes. Specifically, the network visualization component not only serves as the baseline for constructing deeper networks, but also provides a way to intuitively and physically interpret the convergence and optimization of the network. The recursive component enables us to gradually expand the optimization search space as we train deeper and deeper networks to find the best locally optimized model. Taken together, these two components relax the need to train multiple randomly initialized networks in order to achieve good accuracy (in the offline training and online prediction modes) and a well-calibrated model (in the online prediction). Such improvement allows us instead to reliably use shallower, more compact networks with similar accuracy and calibration performance as deeper networks, especially when a reference model to validate does not exist.

While the primary intent of visualizing the network as a unit-cell analog focuses on the explainability of the network, it opens up an interesting aspect for comparison with the original periodic unit cell used for generating the training data. When comparing the DNS results between the original periodic cell and that of the DMN unit-cell analog, we can gauge how this visualization would

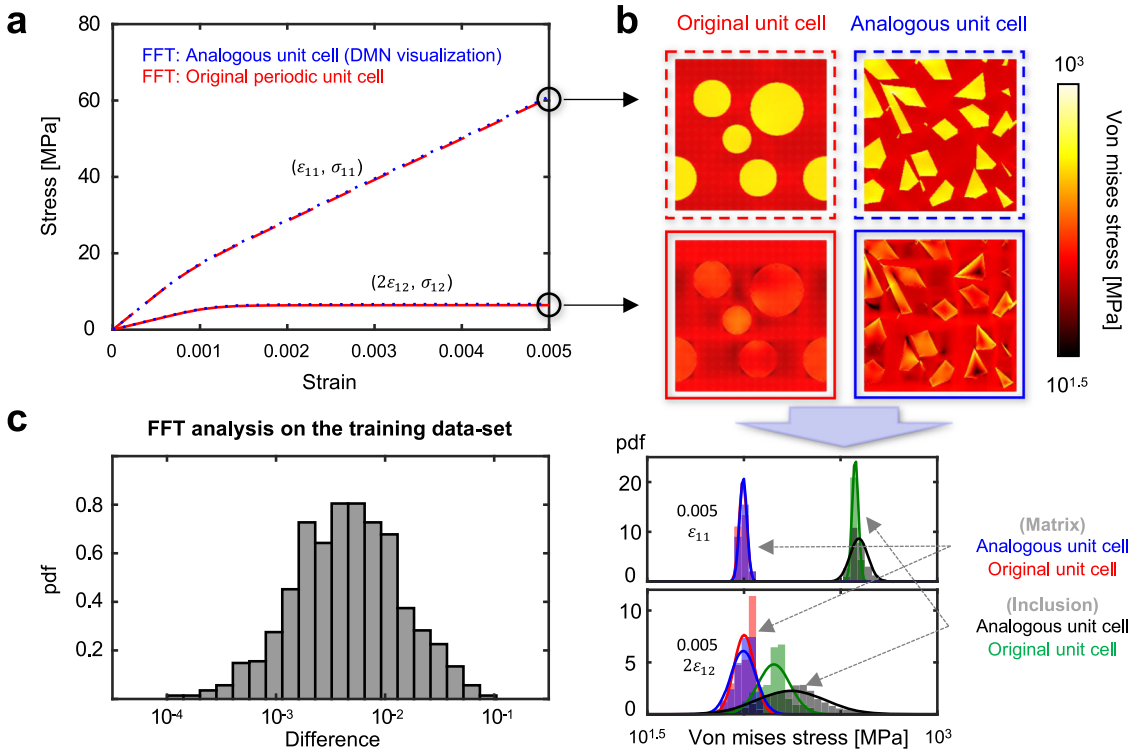

Fig. 8 Interpretation of network visualization when used as unit-cell input in DNS. a Comparison of the macroscopic Norton viscoplastic predictions between the original periodic unit cell and the DMN unit-cell analog. **b** Comparison of the microscopic Norton viscoplastic predictions between the original periodic unit cell and the DMN unit-cell analog. **c** Probability distribution of the difference between FFT-based homogenized elastic constants from the original periodic unit cell and the DMN unit-cell analog.

perform when analyzed as a DNS microstructural realization. In Fig. 8, we show a comparison of the macroscopic stress-strain relationship, the distribution of local stress fields, and the distribution of effective elastic properties when the original periodic unit cell and the DMN unit-cell analog are used as unit-cell inputs directly in DNS and calculated by FFT. Both microstructures have the same phase fraction of inclusions; however, the DMN analogous unit cell contains more inclusions that are smaller and irregular in shape by construction as compared to the original unit cell. Even though these microstructural topologies are different, we observe in Fig. 8a, b that the macroscopic and microscopic responses of both unit cells for two loading configurations yield almost the same results. The distribution of local stresses between the two unit cells is in good agreement except at the tails of the distributions. The distribution of local stresses for the unit-cell analog has longer tails than the one obtained with the original periodic unit cell. This trend is to be expected since the unit-cell representation of the network is, by construction, composed of sharp inclusions leading to stress concentrations at these points. As noted before, the visual representation of the network as a unit cell is not unique. Moving forward, it is possible to consider other visual constructs that would yield even better comparisons. Additionally, we show in Fig. 8c the difference in predicted homogenized elastic stiffness constants when using the original periodic unit cell or the optimized DMN unit-cell analog across the same training dataset used initially to train the model. This difference is calculated as $\sqrt{(||\mathbf{C}_n^{\mathrm{FFT}} - \mathbf{C}_n^{\mathrm{DMN}}||_2)/||\mathbf{C}_n^{\mathrm{FFT}}||_2}$ for all of the n data in our dataset, with $||\cdot||_2$ as the 2-norm value of all matrix components. $\mathbf{C}_n^{\mathrm{FFT}}$ refers to the effective stiffness tensor for data n using the original periodic unit cell as the microstructure for the FFT analysis, and $\mathbf{C}_n^{\mathrm{DMN}}$ refers to the effective stiffness tensor for data n using the optimized DMN unit-cell analog instead. The result shows a difference following a lognormal distribution with a mean value of less than 0.01, emphasizing here again the good agreement between the two microstructures. These results further illustrate that the visualization of the network as an analogous unit cell not only offers a robust methodology to construct and explain the performance of the network, but it also captures how the network constructs the interactions between material phases, to the point where it can reproduce the macroscopic and local responses of the original periodic unit cell, at least for simplistic DNS microstructures (e.g., spherical inclusions).

In this study, we demonstrated our training and visualization approach on binary composite microstructures based on the DMN architecture proposed by Gajek and workers[32] and Nguyen and Noels[34]. Moving forward, we expect that this visualization and recursive training strategy could be extended to other DMN architectures, such as the one by Liu and coworkers[30,31], which allows for the material orientation at the nodes to vary. In this case, this approach could be used for other classes of microstructures, notably polycrystalline materials. After training the DMN, the resulting optimized DMN unit-cell analog could be visualized as a polycrystalline analog (as an example, see Layer 2 in Fig. 3a). It can then be quilted similarly to the protocol presented above to construct a recursive training. Other extensions to non-local constitutive models[53–55] would require not only a modification of the network building blocks themselves, but also additional features to appropriately capture and reflect the intrinsic length scale associated with such models in our visualization and recursive training. Although the approach used in this study could be improved with a better refinement of the visualization of the network, such as the consideration of multiple microstructural length scales or the representation of the network as an actual microstructural representation of the DNS periodic

unit cell, this work is a step towards better strategies for developing explainable DMN with improved accuracy.

METHODS

FFT direct numerical simulations

The training dataset for the offline training mode (elastic homogenization of two-phase composites) and the validation dataset for the online learning mode (elasto-viscoplastic homogenization of two-phase composites with viscoplastic behavior given by Norton's law) were obtained via FFT-based full-field simulations[9–11]. This DNS computational method evaluates the strain and stress fields in periodic media and calculates the effective properties based on averages of those field variables. In this method, the microstructure is discretized into a regular set of materials points or voxels. The micromechanical problem, which consists of simultaneously satisfying the constitutive law (elastic or elasto-viscoplastic) at each voxel, the stress field equilibrium, and strain field compatibility, is solved using FFT. The local problem is reformulated by introducing a homogeneous linear elastic reference medium and a corresponding polarization field, which is a function of the reference medium's stiffness and the a priori unknown stress and strain fields, solutions of the problem. In this way, the solution of the governing equation—a PDE where the unknown is the displacement field—is given by the convolution integral between the Green's function of the displacement field and a body-force derived from the polarization field. With this reformulation, the problem can be advantageously solved, performing the convolution in Fourier space as mere product, and anti-transforming it. The FFT approach yields improved guesses of the unknown stress and strain fields, and thus of the polarization field for the next iteration, and so on and so forth, until convergence is achieved when the input and output stress and strain fields coincide within a small tolerance. The rate of convergence of the method depends on the choice of the reference medium, but the final converged solution does not. In our case, for the two-phase microstructures, we chose our reference medium to have the volumetric average elastic stiffness of the two phases (Voigt average). For the offline training, the constitutive law is based on the linear anisotropic elasticity relation: $\boldsymbol{\sigma} = \mathbf{C}\boldsymbol{\epsilon}$. During the linear homogenization process, the results are obtained based on augmented Lagrangian iterative procedure[56] and the global symmetry of the effective stiffness is not explicitly enforced, yet the FFT analysis basically predicts the correct symmetry of the elastic tensors[11]. For the online prediction results, we adopted an elasto-viscoplastic constitutive relation with viscoplastic behavior given by Norton's law[49,50] such that,

$$\dot{\boldsymbol{\epsilon}} = \dot{\boldsymbol{\epsilon}}^{\mathrm{e}} + \dot{\boldsymbol{\epsilon}}^{\mathrm{p}} = \mathbf{C}^{-1}\dot{\boldsymbol{\sigma}} + \frac{3}{2\sigma_0}\left(\frac{\sigma_{\mathrm{eq}}}{\sigma_0}\right)^{q-1}\boldsymbol{\sigma}, \tag{1}$$

where $\dot{\boldsymbol{\epsilon}}$ is the total strain rate, $\dot{\boldsymbol{\epsilon}}^{\mathrm{e}}$ is the elastic strain rate, $\dot{\boldsymbol{\epsilon}}^{\mathrm{p}}$ is the plastic strain rate, q is a stress exponent, $\sigma_{\mathrm{eq}} = \sqrt{3/2}\cdot||\boldsymbol{\sigma}'||$ is the equivalent stress ($\boldsymbol{\sigma}'$ being the deviatoric stress), and σ_0 is the flow stress. For this constitutive model, we considered the 2D (256×256 voxels) and 3D ($64\times64\times64$ voxels) periodic unit cells illustrated in Figs. 6, 7. Some of the microstructures presented in Fig. 6 (the last two microstructures from the left) were generated using the phase-field method[51,52]. The following materials properties were used for the elasto-viscoplastic two-phase composite. The matrix and inclusions phases were assumed to be elastically isotropic, with Young's moduli 100 and 500 GPa and Poisson's ratios 0.3 and 0.19, respectively, while the flow stress of the matrix was selected to be $\sigma_0 = 100$ MPa, with a stress exponent $q = 10$. The inclusions were assumed to remain linear and elastic. We considered loadings corresponding to uniaxial tension ϵ_{11} and simple shear ϵ_{12} in 2D, and ϵ_{13} in 3D, with an applied strain rate of $1\ \mathrm{s}^{-1}$ along those components.

Training dataset

For a given microstructure (2D or 3D), we generated a training dataset composed of approximately 1000 effective elastic stiffness tensors calculated via FFT-based homogenization. The elastic stiffness tensor ($\mathbf{C}_i$ with $i = 1$ or 2) of each constituent phase was considered to be orthotropic. In 2D, the matrix form for the elastic stiffness tensor is given by,

$$\mathbf{C}_i = \begin{bmatrix} C_i^{11} & C_i^{12} & 0 \\ C_i^{21} & C_i^{22} & 0 \\ 0 & 0 & C_i^{66} \end{bmatrix}, \quad i = 1, 2. \tag{2}$$

In 3D, it is more convenient to express the inverse of the elastic stiffness tensor as a function of the directional Young's moduli E^{ij}, shear moduli G^{ij}, and Poisson's ratios v^{ij}, such that,

$$\mathbf{C}_i^{-1} = \begin{bmatrix} 1/E_i^{11} & -v_i^{21}/E_i^{22} & -v_i^{31}/E_i^{33} & 0 & 0 & 0 \\ -v_{12}/E_i^{11} & 1/E_i^{22} & -v_i^{32}/E_i^{33} & 0 & 0 & 0 \\ -v_{13}/E_i^{11} & -v_i^{23}/E_i^{22} & 1/E_i^{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/G_i^{23} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/G_i^{31} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/G_i^{12} \end{bmatrix}, \quad i = 1, 2. \tag{3}$$

The stiffness tensor of each individual phase was sampled separately via a Latin Hypercube Sampling (LHS) scheme[57]. Following the approach by Nguyen and Noel[34], when sampling the elastic constants individually, we imposed bounding conditions on the range of elastic constants being sampled in order to satisfy the positive definiteness of the elastic stiffness tensors. In 2D, these conditions read:

$$C_1^{11} = 1, \quad \ln(C_2^{11}) \in U(-1, 1), \tag{4}$$

$$\ln(C_i^{22}/C_i^{11}) \in U(-1, 1), \quad C_i^{12}/(C_i^{11}C_i^{22}) \in U(0, 0.9), \quad C_i^{66}/\sqrt{C_i^{11}C_i^{22}} \in U(-1, 1), \quad i = 1, 2, \tag{5}$$

where U(a, b) corresponds to a uniform distribution in the range [a, b]. This 2D-microstructure scheme resulted in sampling 7 parameters for both phases combined.

In 3D, we imposed the positive definiteness conditions by carefully sampling Young's modulus E, shear modulus G, and Poisson's ratio v directly, following the approach by ref. [31]. We started by sampling Young's moduli from a uniform distribution such that, $(E_i^{11}, E_i^{22}, E_i^{33}) \in U(0.1, 10)$, with $i = 1$, 2. We then normalized and scaled Young's moduli such that,

$$\begin{aligned} (E_1^{11}, E_1^{22}, E_1^{33}) &\leftarrow \tfrac{1}{E_1^{11}\cdot E_1^{22}\cdot E_1^{33}}(E_1^{11}, E_1^{22}, E_1^{33}), \\ (E_2^{11}, E_2^{22}, E_2^{33}) &\leftarrow \tfrac{\zeta}{E_2^{11}\cdot E_2^{22}\cdot E_2^{33}}(E_2^{11}, E_2^{22}, E_2^{33}), \end{aligned} \tag{6}$$

where $\zeta \in U(10^{-3}, 10^{3})$ is a random scaling factor. The shear modulus and Poisson's ratio were then sampled within bounds derived from Young's elastic moduli such that,

$$G_i^{12}/\sqrt{E_i^{11}E_i^{22}} \in U(0.25, 0.5), \quad G_i^{23}/\sqrt{E_i^{22}E_i^{33}} \in U(0.25, 0.5), \quad G_i^{13}/\sqrt{E_i^{33}E_i^{11}} \in U(0.25, 0.5), \tag{7}$$

$$v_i^{12}/\sqrt{E_i^{22}/E_i^{11}} \in U(0, 0.5), \quad v_i^{23}/\sqrt{E_i^{33}/E_i^{22}} \in U(0, 0.5), \quad v_i^{31}/\sqrt{E_i^{11}/E_i^{33}} \in U(0, 0.5), \quad i = 1, 2. \tag{8}$$

This 3D-microstructure scheme resulted in sampling 19 parameters for both constituents combined. The wall-clock time to generate all the training data for a given microstructure required around 800 min for 2D and around 13,000 min for 3D on an Apple M1 Max chip.

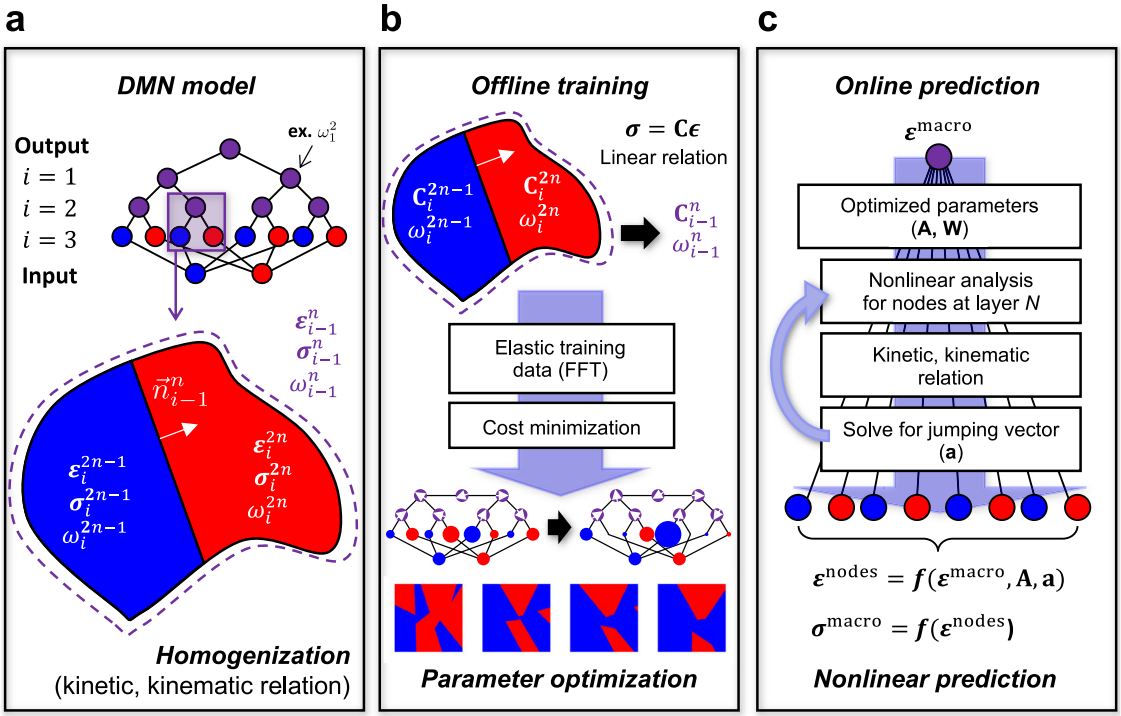

Fig. 9 Schematic description of the deep material network. a Homogenization process in a building block. The parent node at a given layer $i-1$ receives the stress ($\boldsymbol{\sigma}$)/strain (ϵ) information from the child layer i. Weights (ω) and normal vectors ($\vec{n}$) are used to predict the homogenized stress/strain. **b** Offline training procedure. The network is trained on elastic training data only, considering the linear elastic constitutive relationship in the homogenization process (elastic stiffness tensor **C**). The trained network provides the optimized network parameters. **c** Online prediction procedure. The network uses the optimized parameters (in matrix form **A** and **W**) from the offline training. By finding a jumping vector (**a**) satisfying the kinetic/kinematic relation on the homogenized nodes, the network predicts the nonlinear behavior. During the online prediction, the nonlinear constitutive relationship is only considered for nodes in the base layer N.

Kinematic and kinetic relationships for the deep material network building blocks

The building blocks for the deep material network are based on classical continuum homogenization theories[33,34]. This generic homogenization process rests on several kinematic and kinetic relationships between the macroscopic strains and stresses and their microscopic counterparts to derive the homogenized material behavior. The operations at the network nodes use these relations to learn the homogenization pathway, making the deep material network a microstructure-informed/physics-informed network satisfying these relations at each node.

The first of these relationships relate the macroscopic homogenized strains and stresses (in Voigt notation), $\boldsymbol{\epsilon}_{\text{h}}$ and $\boldsymbol{\sigma}_{\text{h}}$, to the microscopic strains and stresses, $\boldsymbol{\epsilon}_i$ and $\boldsymbol{\sigma}_i$, of phases i such that,

$$\boldsymbol{\epsilon}_{\text{h}}=\int \boldsymbol{\epsilon} dV=\sum_i \boldsymbol{\epsilon}_i f_i,\quad \boldsymbol{\sigma}_{\text{h}}=\int \boldsymbol{\sigma} dV=\Sigma \boldsymbol{\sigma}_i f_i,\quad \text{and}\quad \sum_i f_i=1. \tag{9}$$

The second relationship pertains to the Hill–Mandel condition[58] for energy conservation. This relationship reads,

$$\boldsymbol{\epsilon}_{\text{h}}^{\text{T}}\boldsymbol{\sigma}_{\text{h}}=\int \boldsymbol{\epsilon}^{\text{T}}\boldsymbol{\sigma} dV=\sum_i \boldsymbol{\epsilon}_i^{\text{T}}\boldsymbol{\sigma}_i f_i. \tag{10}$$

When only two phases are considered for each building block, the third relation accounts for the continuity of traction across the interface between the two phases such that,

$$\mathbf{H}^{\text{T}}(\boldsymbol{\sigma}_2-\boldsymbol{\sigma}_1)=\mathbf{0}, \tag{11}$$

where the **H** matrix is an orientation matrix defined for a given normal vector, $\vec{n}_{2\text{D}}=[\cos 2\pi\theta, \sin 2\pi\theta]$ in 2D, and $\vec{n}_{3\text{D}}=[\cos 2\pi\theta \sin \pi\phi, \sin 2\pi\theta \sin \pi\phi, \cos \pi\phi]$ in 3D, such that,

$$\mathbf{H}(\vec{n}_{2\text{D}})=\begin{bmatrix}\vec{n}(1) & 0\\ 0 & \vec{n}(2)\\ \vec{n}(2) & \vec{n}(1)\end{bmatrix},\quad \mathbf{H}(\vec{n}_{3\text{D}})=\begin{bmatrix}\vec{n}(1) & 0 & 0\\ 0 & \vec{n}(2) & 0\\ 0 & 0 & \vec{n}(3)\\ 0 & \vec{n}(3) & \vec{n}(2)\\ \vec{n}(3) & 0 & \vec{n}(1)\\ \vec{n}(2) & \vec{n}(1) & 0\end{bmatrix}. \tag{12}$$

Note that the traction continuity condition in Eq. (11) can be extended to the more general case for multi-phase composites[32,34]. Combining Eq. (9) with the Hill–Mandel Eq. (10) yields,

$$(\boldsymbol{\epsilon}_1^{\text{T}}-\boldsymbol{\epsilon}_2^{\text{T}})(\boldsymbol{\sigma}_1-\boldsymbol{\sigma}_2)=0, \tag{13}$$

which provides an additional condition on the strain when considering the traction condition in Eq. (11) such that,

$$(\boldsymbol{\epsilon}_1-\boldsymbol{\epsilon}_2)^{\text{T}}=\frac{1}{f_1 f_2}(\mathbf{Hb})^{\text{T}}, \tag{14}$$

for an arbitrary vector **b**. The relationship derived in Eq. (14) articulates a correlation between the microscopic strains via the normal vector. Taking advantage of Eq. (9), we can finally express the local strains as a function of the homogenized strain such that,

$$\boldsymbol{\epsilon}_1=\boldsymbol{\epsilon}_{\text{h}}+\frac{1}{f_1}\mathbf{Hb},\quad \boldsymbol{\epsilon}_2=\boldsymbol{\epsilon}_{\text{h}}-\frac{1}{f_2}\mathbf{Hb}. \tag{15}$$

Relationships in Eqs. (9)–(15) serve as the basis for the architecture of the deep material network and are used both during offline training and online prediction. The illustration of this homogenization process for a given building block with layer notations can be found in Fig. 9a.

Deep material network architecture

As shown in Fig. 9a, the deep material network architecture consists of a binary tree-type network. In this architecture, at a given layer, pairs of (child) nodes are merged via the homogenization process into one (parent) node in the next layer. For a network of depth N (Fig. 9a illustrates a rudimentary network with $N=3$), the last N-th layer is composed of 2^N nodes with 2^N activation weights $(\omega_N^1, \ldots, \omega_N^{2N})$. The other $N-1$ layers are composed of inherited weights from the previous layers $(\omega_i^n = \omega_{i+1}^{2n-1} + \omega_{i+1}^{2n})$ and also of normal vectors $\vec{n}_i^n$. Here the indices i and n refer to the i-th layer and the n-th node within that layer, respectively.

In 2D, this construct results in $2^{N+1}-1$ network parameters (2^N base-layer weights in the base layer and 2^N-1 parameters for the normal vectors in the remaining $N-1$ layers) to be optimized during the offline training mode. In 3D, there are $3\cdot 2^N-2$ network parameters (2^N base-layer weights in the base layer and $2^{N+1}-2$ parameters (spherical coordinates) for the normal vectors in the remaining $N-1$ layers). We used a rectified linear activation function[59], or ReLU, for the base-layer weights, since negative weights do not have a physical meaning. As a result, all the activation weights have non-negative values. For the parameter initialization, all network components were set between 0 and 1, and the weights were normalized, to sum up to one. We used PyTorch[60] libraries for the network's forward and back propagation implementation.

This architecture provides a natural connection between the nodal information contained in the base layer N and the final output node in Layer 0. This assertion is especially true and useful for expressing the nodal strains as a function of the strain in node 0 based on the kinematic homogenization condition expressed in Eq. (15). Indeed, the nodal strain information in the base layer N ($\boldsymbol{\epsilon}^{\text{nodes}} = \boldsymbol{\epsilon}_N^1, \ldots, \boldsymbol{\epsilon}_N^{2^N}$) can be linearly correlated to the final microstructure's homogenized macroscopic strain $\boldsymbol{\epsilon}^{\text{macro}} = \boldsymbol{\epsilon}_0^1$, where 0 denotes the output layer and 1 the only node contained in that layer. If we know the **b** vector satisfying the kinetic relation in Eq. (11), this linear correlation can be expressed in terms of the weights, the rotation matrix **H**, and the **b** vector such that,

$$\boldsymbol{\epsilon}_N^n = \boldsymbol{\epsilon}^{\text{macro}} + \sum_{i=1}^{N}\left(\frac{\mathbf{H}(\vec{n}_{i-1}^{\,1+\alpha_{N-i+1}^n})}{\omega_i^{1+\alpha_{N-i}^n}}(-1)^{\beta_{N-i}^n}\right)\left(\omega_{i-1}^{1+\alpha_{N-i+1}^n}\mathbf{b}_{i-1}^{1+\alpha_{N-i+1}^n}\right), \quad \text{with} \quad (\alpha_i^n, \beta_i^n) = \text{divmod}\,(n-1, 2^i). \tag{16}$$

By defining $\boldsymbol{\epsilon}^{\text{local}}$ as the vector containing the local strain perturbation in the base layer, we can express it as,

$$\boldsymbol{\epsilon}^{\text{local}} = \begin{bmatrix}\boldsymbol{\epsilon}_N^1\\ \boldsymbol{\epsilon}_N^2\\ \vdots\\ \boldsymbol{\epsilon}_N^{2^N}\end{bmatrix} - \begin{bmatrix}\boldsymbol{\epsilon}^{\text{macro}}\\ \boldsymbol{\epsilon}^{\text{macro}}\\ \vdots\\ \boldsymbol{\epsilon}^{\text{macro}}\end{bmatrix} = \begin{bmatrix}\mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \ldots & \mathbf{A}_{1,2^N-1}\\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} & \ldots & \mathbf{A}_{2,2^N-1}\\ \vdots & \vdots & \vdots & \vdots\\ \mathbf{A}_{2^N,1} & \hat{\mathbf{A}}_{2^N,2} & \ldots & \mathbf{A}_{2^N,2^N-1}\end{bmatrix}\begin{bmatrix}\mathbf{a}_1\\ \mathbf{a}_2\\ \vdots\\ \mathbf{a}_{2^N-1}\end{bmatrix} = \mathbf{A}\mathbf{a}, \tag{17}$$

where $i=1, \ldots, N$, $n=1, \ldots, 2^N$, and

$$\begin{aligned}\mathbf{a}_{2^{i-1}+j-1} &= \omega_{i-1}^j\mathbf{b}_{i-1}^j && \text{for } j=1,\ldots,2^{i-1},\\ \mathbf{A}_{n,2^{i-1}+j-1} &= \left(\frac{\mathbf{H}(\vec{n}_{i-1}^{\,1+\alpha_{N-i+1}^n})}{\omega_i^{1+\alpha_{N-i}^n}}\right)(-1)^{\beta_{N-i}^n} && \text{for } j=1+\alpha_{N-i+1}^n,\\ \mathbf{A}_{n,2^{i-1}+j-1} &= \mathbf{0}_{6\times 3} && \text{for } j\neq 1+\alpha_{N-i+1}^n.\end{aligned} \tag{18}$$

The vector **a** corresponds to a jumping (displacement) vector, and **A** is a weighted rotation matrix representing the DMN architecture in matrix form. The expression in Eq. (17) is especially useful in the online prediction to relate the nodal strain information to the pre-trained network parameters and the microstructure's homogenized macroscopic strain information.

Offline training mode

As shown in Fig. 9b, in the offline training mode, the network is trained only on elastic properties to optimize the network's parameters. In this case, the network takes in as inputs the elastic stiffness tensors of the individual phases composing the microstructure and outputs the effective elastic stiffness tensor of that microstructure. The training is based on a limited database of effective elastic stiffness tensors calculated by FFT to train and optimize the network parameters, composed of the base-layer weights in the base layer N and the normal vectors in the remaining $N-1$ layers. At a given layer $i-1$, each node receives the elastic properties $(\mathbf{C}_i^{2n-1}, \mathbf{C}_i^{2n})$ and activation weights $(\omega_i^{2n-1}, \omega_i^{2n})$ from the nodes in layer i. Along with the normal vector in the node at the current layer $i-1$ ($\vec{n}_{i-1}^n$), the network can be used to derive the homogenized elastic properties in the node ($\mathbf{C}_{i-1}^n$). Making use of the simple relationship between phase fractions and weights ($f_i^{2n-1} = \omega_i^{2n-1}/(\omega_i^{2n-1}+\omega_i^{2n})$) and by considering the linear constitutive relation of the child nodes ($\boldsymbol{\sigma}_i^{2n-1} = \mathbf{C}_i^{2n-1}\boldsymbol{\epsilon}_i^{2n-1}$ and $\boldsymbol{\sigma}_i^{2n} = \mathbf{C}_i^{2n}\boldsymbol{\epsilon}_i^{2n}$) combined with the continuity of traction across the interface in Eq. (11) and the relations between the local strains and the homogenized strain in Eq. (15), we obtain a linear relation between the **b** vector and the homogenized strain such that,

$$\begin{aligned}\mathbf{b}_{i-1}^n &= \mathbf{B}_{i-1}^n\boldsymbol{\epsilon}_{i-1}^n\\ &= -f_i^{2n-1}f_i^{2n}\left(\mathbf{H}\left(\vec{n}_{i-1}^n\right)^{\mathrm{T}}(f_i^{2n-1}\mathbf{C}_i^{2n}+f_i^{2n}\mathbf{C}_i^{2n-1})\mathbf{H}\left(\vec{n}_{i-1}^n\right)\right)^{-1}\\ &\quad \mathbf{H}\left(\vec{n}_{i-1}^n\right)^{\mathrm{T}}(\mathbf{C}_i^{2n-1}-\mathbf{C}_i^{2n})\boldsymbol{\epsilon}_{i-1}^n.\end{aligned} \tag{19}$$

Using the **B** matrix in Eq. (19) and the stress homogenization in Eq. (9) with the linear constitutive relation of the parent node ($\boldsymbol{\sigma}_{i-1}^n = \mathbf{C}_{i-1}^n\boldsymbol{\epsilon}_{i-1}^n$), we can relate the stress and strain at a given layer $i-1$ as a function of the phase fraction and elastic stiffness tensors from the child nodes and the normal vector at that node:

$$\begin{aligned}\boldsymbol{\sigma}_{i-1}^n = \mathbf{C}_{i-1}^n\boldsymbol{\epsilon}_{i-1}^n = \Big[&f_i^{2n-1}\mathbf{C}_i^{2n-1}\left(\mathbf{1}+\tfrac{1}{f_i^{2n-1}}\mathbf{H}\left(\vec{n}_{i-1}^n\right)\mathbf{B}_{i-1}^n\right)\\ &+f_i^{2n}\mathbf{C}_i^{2n}\left(\mathbf{1}-\tfrac{1}{f_i^{2n}}\mathbf{H}\left(\vec{n}_{i-1}^n\right)\mathbf{B}_{i-1}^n\right)\Big]\boldsymbol{\epsilon}_{i-1}^n.\end{aligned} \tag{20}$$

In the offline training mode, the node information is propagated forward from every pair of nodes in the child layer to nodes in the parent layer. This forward propagation allows us to train the network by optimizing the weights in the base layer N and the normal vector at each node for all the remaining $N-1$ layers. Activation weights in layer 0 to layer $N-1$ are inherited from weights in the base layer N by adding the weight of the child nodes. To train the network, we minimize the cost function defined as:

$$\text{Cost func.} = 1000\cdot\left[\sum_{n=1}^{2^N}\text{ReLU}(\omega_N^n)-1\right]^2 + \frac{1}{N_{\text{batch}}}\sqrt{\sum_{n=1}^{N_{\text{batch}}}\frac{||\mathbf{C}_n^{\text{FFT}}-\mathbf{C}_n^{\text{DMN}}||_2}{||\mathbf{C}_n^{\text{FFT}}||_2}}, \tag{21}$$

where $\mathbf{C}_n^{\text{FFT}}$ and $\mathbf{C}_n^{\text{DMN}}$ are the effective stiffness matrices obtained by FFT and predicted by the network, respectively, $||\cdot||_2$ refers to the 2-norm value of all matrix components, and N_{batch} is the batch size. The first term in Eq. (21) represents a constraint on the weights of a given layer summing up to 1, while the second term compares the predicted elastic stiffness tensor from the network to that of the training data.

To train the network, we used a batch size of $N_{\text{batch}}=20$ with the ADAM optimizer[61] and the AMSGrad method[62] for the cost function in Eq. (21), using an adaptive learning rate set to $0.5\cdot 10^{-4}\cdot(1+\cos(10\pi m/M))$, where m is the current epoch number and M is the total number of epochs[63]. To be consistent in our comparison between random initialization and recursive training, we fixed M to be 4 000. We trained our network on 80% of the elastic stiffness data calculated via FFT, and the remaining 20% was used as a validation set. In terms of computational cost, training the DMN for

N= 3, 5, and 7 in 2D took approximately 12, 16, and 27 minutes, $N = 4$ and 7 in 3D took approximately 20 and 25 min, respectively on an Apple M1 Max chip. Subsequent evaluation of the network to predict the homogenized stiffness matrix for a given composite took approximately a few milliseconds, showing a dramatic performance in computational cost.

Online prediction mode

The online prediction mode does not need to be trained on pre-existing data as it is based on the weights and normal vectors determined from the offline training mode. Instead, as illustrated in Fig. 9c, the network takes in the macroscopic strain as an input and makes use of the network architecture linking the macroscopic strain to the nodal strain in Eq. (17). By using the kinetic and energetic constraints in Eqs. (11) and (15) for all the nodes, the network iteratively solves the (nonlinear) stress-strain response given a (nonlinear) constitutive relationship for the base nodes at layer N. In this study, we used the elasto-viscoplastic constitutive relation with viscoplastic behavior given by Norton's law given by Eq. (1). Materials properties and loading conditions were identical to those used in our FFT-based simulations.

considered as degrees of freedom since the $\mathbf{A}$ matrix in Eq. (18) cannot be set, since it physically means the strain difference and residual force cannot be defined in this case.

The residual vector $\mathbf{R}$ represents the residual forces acting at each homogenized node and can be written as,

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ \vdots \\ \mathbf{R}_{2^N-1} \end{bmatrix} = \begin{bmatrix} \mathbf{H}\left(\overrightarrow{n}_0^1\right)^{\mathrm{T}}\left(\boldsymbol{\sigma}_1^1 - \boldsymbol{\sigma}_1^2\right) \\ \mathbf{H}\left(\overrightarrow{n}_1^1\right)^{\mathrm{T}}\left(\boldsymbol{\sigma}_2^1 - \boldsymbol{\sigma}_2^2\right) \\ \vdots \\ \mathbf{H}\left(\overrightarrow{n}_{N-1}^{2^{N-1}}\right)^{\mathrm{T}}\left(\boldsymbol{\sigma}_N^{2^N-1} - \boldsymbol{\sigma}_N^{2^N}\right) \end{bmatrix}. \tag{22}$$

Note that all the components of the $\mathbf{R}$ vector can be expressed as a function of the stress at layer N following Eq. (9). For example, $\mathbf{R}_1$ can be expressed as

$$\mathbf{R}_1 = \mathbf{H}(\overrightarrow{n}_0^1)^{\mathrm{T}}\left(\sum_{n=1}^{2^{N-1}} f_N^n \sigma_N^n - \sum_{n=2^{N-1}+1}^{2^N} f_N^n \sigma_N^n\right). \tag{23}$$

Algorithm 2. DMN online implementation for the Norton viscoplastic analysis

Input : Final macroscopic strain ($\boldsymbol{\varepsilon}^{\mathrm{total}}$), total time ($t^{\mathrm{total}}$)
Output : $\boldsymbol{\sigma}^{\mathrm{macro}}(t) - \boldsymbol{\varepsilon}^{\mathrm{macro}}(t)$

Initialize time, macroscopic strain, jumping vector ($t = 0$, $\boldsymbol{\varepsilon}^{\mathrm{macro}} = \mathbf{0}, \mathbf{a} = \mathbf{0}$)
Construct the DMN model's $\mathbf{A}, \mathbf{W}$ matrices
Set the time increment ($\delta_t = t^{\mathrm{total}}/100$)
while $t \leq t^{\mathrm{total}}$ **do**
  Initialize Residual ($\mathbf{R}$)
  Update time, macroscopic strain: $t = t + \delta_t$, $\boldsymbol{\varepsilon}^{\mathrm{macro}}(t) = \boldsymbol{\varepsilon}^{\mathrm{total}} \cdot t/t^{\mathrm{total}}$
  while $||\mathbf{R}|| < \mathrm{Tol}(= 10^{-8})$ **do**
    Calculate local strains: $\boldsymbol{\varepsilon}^{\mathrm{local}} = \mathbf{A}\mathbf{a}$
    Update strain at the base nodes: $\boldsymbol{\varepsilon}^{\mathrm{nodes}} = \boldsymbol{\varepsilon}^{\mathrm{macro}} + \boldsymbol{\varepsilon}^{\mathrm{local}}$
    Calculate stress/tangent stiffness at the base nodes using Norton constitutive relationship: $[\boldsymbol{\sigma}^{\mathrm{nodes}}, \mathbf{K}] = \mathrm{function}(\boldsymbol{\varepsilon}^{\mathrm{nodes}}(t), \boldsymbol{\varepsilon}^{\mathrm{nodes}}(t - \delta_t))$
    Evaluate residual: $\mathbf{R} = \mathbf{A}^{\mathrm{T}}\mathbf{W}\boldsymbol{\sigma}^{\mathrm{nodes}}$
    Evaluate Jacobian: $\mathbf{J} = \mathbf{A}^{\mathrm{T}}\mathbf{W}\mathbf{K}\mathbf{A}$
    Calculate increment in jumping vector: $\delta\mathbf{a} = -\mathbf{J}^{-1}\mathbf{R}$
    Update jumping vector: $\mathbf{a} = \mathbf{a} + \delta\mathbf{a}$
  end
  Calculate macroscopic stress: $\boldsymbol{\sigma}^{\mathrm{macro}}(t) = \sum_{j=1}^{2^N} \omega_N^j \boldsymbol{\sigma}_N^j$
end

The pseudo-code detailing the iterative implementation is listed in Algorithm 2 using the Newton–Raphson method[64]. For a given macroscopic strain $\epsilon^{\mathrm{macro}}(t)$ at time t, the algorithm computes the corresponding macroscopic stress $\boldsymbol{\sigma}^{\mathrm{macro}}(t)$ by minimizing a residual $\mathbf{R}$. As the constitutive relation changes, the jumping vector in Eq. (17) is unknown, such that the kinetic conditions (Eq. (11)) in $N-1$ layers should be satisfied by obtaining the jumping vector, $\mathbf{a}$, numerically. This procedure calculates stress and tangent stiffness at each base node following the elasto-viscoplastic constitutive relation with Norton's viscoplastic behavior. Once the macroscopic stress is calculated at a given time t, the macroscopic strain is incremented until it reaches the final, preset macroscopic strain. The nodes with a null weight are not

Hence, the residual vector can be expressed as,

$$\mathbf{R} = \mathbf{A}^{\mathrm{T}}\mathbf{W}\begin{bmatrix} \boldsymbol{\sigma}_N^1 \\ \boldsymbol{\sigma}_N^2 \\ \vdots \\ \boldsymbol{\sigma}_N^{2^N} \end{bmatrix} = \mathbf{A}^{\mathrm{T}}\mathbf{W}\boldsymbol{\sigma}^{\mathrm{nodes}},$$

$$\text{with} \quad \mathbf{W} = \begin{bmatrix} \mathbf{W}_1 & \mathbf{0}_{6\times6} & \dots & \mathbf{0}_{6\times6} \\ \mathbf{0}_{6\times6} & \mathbf{W}_2 & \dots & \mathbf{0}_{6\times6} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0}_{6\times3} & \mathbf{0}_{6\times6} & \dots & \mathbf{W}_{2^N} \end{bmatrix}, \quad \text{and} \quad \mathbf{W}_j = \omega_N^j \mathbf{I}_{6\times6}. \tag{24}$$

Finally, when we define the tangent stiffness of each of the nodes ($\mathbf{K}_n = \partial \boldsymbol{\sigma}_N^n / \partial \boldsymbol{\epsilon}_N^n$) using Eq. (1), and construct a global tangent stiffness matrix $\mathbf{K}$ that we use to evaluate the Jacobian matrix $\mathbf{J} = \mathbf{A}^T\mathbf{W}\mathbf{K}\mathbf{A}$ to update jumping vector. The matrix $\mathbf{K}$ is simply given by,

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_1 & \mathbf{0}_{6\times 6} & \dots & \mathbf{0}_{6\times 6} \\ \mathbf{0}_{6\times 6} & \mathbf{K}_2 & \dots & \mathbf{0}_{6\times 6} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0}_{6\times 3} & \mathbf{0}_{6\times 6} & \dots & \mathbf{K}_{2^N} \end{bmatrix}. \tag{25}$$

Regarding the computational cost of the online prediction, $N = 3$, 5, and 7 recursive DMN in Fig. 4 take about 1.4, 5.2, and 7.2 s for a calculation, respectively, as compared to approximately 1600 s by FFT for 100 time steps on an Apple M1 Max chip. Similar derivations of the DMN and related equations can be found in ref. [33] and in ref. [34].

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request as part of the CINT user program.

CODE AVAILABILITY

The codes used to calculate the results of this study are available from the corresponding author upon reasonable request as part of the CINT user program.

Received: 16 March 2023; Accepted: 9 July 2023;
Published online: 25 July 2023

REFERENCES

1. Capuano, G. & Rimoli, J. J. Smart finite elements: a novel machine learning application. *Comput. Methods Appl. Mech. Eng.* **345**, 363–381 (2019).
2. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707 (2019).
3. Lu, L., Jin, P., Pang, G., Zhang, Z. & Karniadakis, G. E. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nat. Mach. Intell.* **3**, 218–229 (2021).
4. Montes de Oca Zapiain, D., Stewart, J. A. & Dingreville, R. Accelerating phase-field-based microstructure evolution predictions via surrogate models trained by machine learning methods. *npj Comput. Mater.* **7**, 3 (2021).
5. Cuomo, S. et al. Scientific machine learning through physics–informed neural networks: where we are and what's next. *J. Sci. Comput.* **92**, 88 (2022).
6. Hu, C., Martin, S. & Dingreville, R. Accelerating phase-field predictions via recurrent neural networks learning the microstructure evolution in latent space. *Comput. Methods Appl. Mech. Eng.* **397**, 115128 (2022).
7. Oommen, V., Shukla, K., Goswami, S., Dingreville, R. & Karniadakis, G. E. Learning two-phase microstructure evolution using neural operators and autoencoder architectures. *npj Comput. Mater.* **8**, 190 (2022).
8. Feyel, F. & Chaboche, J.-L. FE2 multiscale approach for modelling the elasto-viscoplastic behaviour of long fibre SiC/Ti composite materials. *Comput. Methods Appl. Mech. Eng.* **183**, 309–330 (2000).
9. Moulinec, H. & Suquet, P. A numerical method for computing the overall response of nonlinear composites with complex microstructure. *Comput. Methods Appl. Mech. Eng.* **157**, 69–94 (1998).
10. Michel, J.-C., Moulinec, H. & Suquet, P. Effective properties of composite materials with periodic microstructure: a computational approach. *Comput. Methods Appl. Mech. Eng.* **172**, 109–143 (1999).
11. Lebensohn, R. A., Kanjarla, A. K. & Eisenlohr, P. An elasto-viscoplastic formulation based on fast fourier transforms for the prediction of micromechanical fields in polycrystalline materials. *Int. J. Plast.* **32**, 59–69 (2012).
12. Rastkar, S., Zahedi, M., Korolev, I. & Agarwal, A. A meshfree approach for homogenization of mechanical properties of heterogeneous materials. *Eng. Anal. Bound. Elem.* **75**, 79–88 (2017).
13. Conti, S., Müller, S. & Ortiz, M. Data-driven finite elasticity. *Arch. Ration. Mech. Anal.* **237**, 1–33 (2020).
14. Minh Nguyen-Thanh, V., Trong Khiem Nguyen, L., Rabczuk, T. & Zhuang, X. A surrogate model for computational homogenization of elastostatics at finite strain using high-dimensional model representation-based neural network. *Int. J. Numer. Methods Eng.* **121**, 4811–4842 (2020).
15. Oliver, J., Caicedo, M., Huespe, A. E., Hernández, J. & Roubin, E. Reduced order modeling strategies for computational multiscale fracture. *Comput. Methods Appl. Mech. Eng.* **313**, 560–595 (2017).
16. Mozaffar, M. et al. Deep learning predicts path-dependent plasticity. *Proc. Natl Acad. Sci. USA* **116**, 26414–26420 (2019).
17. Ferreira, B. P., Pires, F. M. A. & Bessa, M. A. Adaptivity for clustering-based reduced-order modeling of localized history-dependent phenomena. *Comput. Methods Appl. Mech. Eng.* **393**, 114726 (2022).
18. Yang, H., Qiu, H., Xiang, Q., Tang, S. & Guo, X. Exploring elastoplastic constitutive law of microstructured materials through artificial neural network - A mechanistic-based data-driven approach. *J. Appl. Mech.* **87**, 091005 (2020).
19. Ford, E., Maneparambil, K., Rajan, S. & Neithalath, N. Machine learning-based accelerated property prediction of two-phase materials using microstructural descriptors and finite element analysis. *Comput. Mater. Sci.* **191**, 110328 (2021).
20. Masi, F. & Stefanou, I. Multiscale modeling of inelastic materials with thermodynamics-based artificial neural networks (TANN). *Comput. Methods Appl. Mech. Eng.* **398**, 115190 (2022).
21. Li, X. et al. A transfer learning approach for microstructure reconstruction and structure-property predictions. *Sci. Rep.* **8**, 13461 (2018).
22. Rao, C. & Liu, Y. Three-dimensional convolutional neural network (3D-CNN) for heterogeneous material homogenization. *Comput. Mater. Sci.* **184**, 109850 (2020).
23. Liu, Z., Bessa, M. A. & Liu, W. K. Self-consistent clustering analysis: an efficient multi-scale scheme for inelastic heterogeneous materials. *Comput. Methods Appl. Mech. Eng.* **306**, 319–341 (2016).
24. Bessa, M. A. et al. A framework for data-driven analysis of materials under uncertainty: countering the curse of dimensionality. *Comput. Methods Appl. Mech. Eng.* **320**, 633–667 (2017).
25. Liu, Z., Fleming, M. & Liu, W. K. Microstructural material database for self-consistent clustering analysis of elastoplastic strain softening materials. *Comput. Methods Appl. Mech. Eng.* **330**, 547–577 (2018).
26. Marshall, A. & Kalidindi, S. R. Autonomous development of a machine-learning model for the plastic response of two-phase composites from micromechanical finite element models. *JOM* **73**, 2085–2095 (2021).
27. Fuhg, J. N., Marino, M. & Bouklas, N. Local approximate Gaussian process regression for data-driven constitutive models: development and comparison with neural networks. *Comput. Methods Appl. Mech. Eng.* **388**, 114217 (2022).
28. Teichert, G. H. & Garikipati, K. Machine learning materials physics: surrogate optimization and multi-fidelity algorithms predict precipitate morphology in an alternative to phase field dynamics. *Comput. Methods Appl. Mech. Eng.* **344**, 666–693 (2019).
29. Hashemi, S. & Kalidindi, S. R. A machine learning framework for the temporal evolution of microstructure during static recrystallization of polycrystalline materials simulated by cellular automaton. *Comput. Mater. Sci.* **188**, 110132 (2021).
30. Liu, Z., Wu, C. T. & Koishi, M. A deep material network for multiscale topology learning and accelerated nonlinear modeling of heterogeneous materials. *Comput. Methods Appl. Mech. Eng.* **345**, 1138–1168 (2019).
31. Liu, Z. & Wu, C. T. Exploring the 3D architectures of deep material network in data-driven multiscale mechanics. *J. Mech. Phys. Solids* **127**, 20–46 (2019).
32. Gajek, S., Schneider, M. & Böhlke, T. An FE–DMN method for the multiscale analysis of short fiber reinforced plastic components. *Comput. Methods Appl. Mech. Eng.* **384**, 113952 (2021).
33. Gajek, S., Schneider, M. & Böhlke, T. On the micromechanics of deep material networks. *J. Mech. Phys. Solids* **142**, 103984 (2020).
34. Nguyen, V. D. & Noels, L. Micromechanics-based material networks revisited from the interaction viewpoint; robust and efficient implementation for multi-phase composites. *Eur. J. Mech. A Solids* **91**, 104384 (2022).
35. Huang, T., Liu, Z., Wu, C. T. & Chen, W. Microstructure-guided deep material network for rapid nonlinear material modeling and uncertainty quantification. *Comput. Methods Appl. Mech. Eng.* **398**, 115197 (2022).
36. Wu, L., Adam, L. & Noels, L. Micro-mechanics and data-driven based reduced order models for multi-scale analyses of woven composites. *Compos. Struct.* **270**, 114058 (2021).
37. Nguyen, V. D. & Noels, L. Interaction-based material network: a general framework for (porous) microstructured materials. *Comput. Methods Appl. Mech. Eng.* **389**, 114300 (2022).
38. Gajek, S., Schneider, M. & Böhlke, T. An FE-DMN method for the multiscale analysis of thermomechanical composites. *Comput. Mech.* **69**, 1087–1113 (2022).
39. Liu, Z. Deep material network with cohesive layers: multi-stage training and interfacial failure analysis. *Comput. Methods Appl. Mech. Eng.* **363**, 112913 (2020).

40. Beniwal, D. & Ray, P. Learning phase selection and assemblages in high-entropy alloys through a stochastic ensemble-averaging model. *Comput. Mater. Sci.* **197**, 110647 (2021).
41. Scardapane, S. & Wang, D. Randomness in neural networks: an overview. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **7**, e1200 (2017).
42. Guo, C., Pleiss, G., Sun, Y. & Weinberger, K. Q. On calibration of modern neural networks. In *International Conference on Machine Learning* 1321–1330 (PMLR, 2017).
43. Frankle, J. & Carbin, M. The lottery ticket hypothesis: finding sparse, trainable neural networks. Preprint at arXiv:1803.03635 (2018).
44. Minderer, M. et al. Revisiting the calibration of modern neural networks. *Adv. Neural Inf. Process. Syst.* **34**, 15682–15694 (2021).
45. Qin, Z., Yu, F., Liu, C. & Chen, X. How convolutional neural network see the world-A survey of convolutional neural network visualization methods. *Math. Found. Comput.* **1**, 149–180 (2018).
46. Gao, T. & Zhang, W. A mass constraint formulation for structural topology optimization with multiphase materials. *Int. J. Numer. Methods Eng.* **88**, 774–796 (2011).
47. Nguyen, L. M., Liu, J., Scheinberg, K. & Takáč, M. SARAH: a novel method for machine learning problems using stochastic recursive gradient. In *International Conference on Machine Learning* 2613–2621 (PMLR, 2017).
48. Zhang, W., Zhang, F., Zhang, J., Zhang, J. & Zhang, J. Optimization of identification structure parameters based on recursive maximum likelihood iteration. In *2018 International Computers, Signals and Systems Conference (ICOMSSC)* 726–731 (IEEE, 2018).
49. Irgens, F. *Continuum Mechanics* (Springer Science & Business Media, 2008).
50. Lebensohn, R. A. et al. Modeling void growth in polycrystalline materials. *Acta Mater.* **61**, 6918–6932 (2013).
51. Stewart, J. A. & Dingreville, R. Microstructure morphology and concentration modulation of nanocomposite thin-films during simulated physical vapor deposition. *Acta Mater.* **188**, 181–191 (2020).
52. Dingreville, R., Stewart, J. A., Chen, E. Y. & Monti, J. M. *Benchmark Problems for the Mesoscale Multiphysics Phase Field Simulator* (*MEMPHIS*). Report no. SAND2020-12852 (Sandia National Laboratories, 2020).
53. Drugan, W. J. & Willis, J. R. A micromechanics-based nonlocal constitutive equation and estimates of representative volume element size for elastic composites. *J. Mech. Phys. Solids* **44**, 497–524 (1996).
54. Dingreville, R., Robbins, J. & Voth, T. E. Wave propagation and dispersion in elasto-plastic microstructured materials. *Int. J. Solids Struct.* **51**, 2226–2237 (2014).
55. Alberdi, R., Robbins, J., Walsh, T. & Dingreville, R. Exploring wave propagation in heterogeneous metastructures using the relaxed micromorphic model. *J. Mech. Phys. Solids* **155**, 104540 (2021).
56. Michel, J., Moulinec, H. & Suquet, P. A computational scheme for linear and nonlinear composites with arbitrary phase contrast. *Int. J. Numer. Methods Eng.* **52**, 139–160 (2001).
57. Stein, M. Large sample properties of simulations using Latin hypercube sampling. *Technometrics* **29**, 143–151 (1987).
58. Hill, R. The elastic behaviour of a crystalline aggregate. *Proc. Phys. Soc. A* **65**, 349 (1952).
59. Agarap, A. F. Deep learning using rectified linear units (ReLU). Preprint at arXiv:1803.08375 (2018).
60. Paszke, A. et al. Pytorch: an imperative style, high-performance deep learning library. *Adv. Neural Inf. Process Syst.* **32** (2019).
61. Kingma, D. P. & Ba, J. Adam: a method for stochastic optimization. Preprint at arXiv:1412.6980 (2014).
62. Reddi, S. J., Kale, S. & Kumar, S. On the convergence of Adam and beyond. Preprint at arXiv:1904.09237 (2019).
63. Loshchilov, I. & Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. Preprint at arXiv:1608.03983 (2016).
64. Ypma, T. J. Historical development of the Newton–Raphson method. *SIAM Rev.* **37**, 531–551 (1995).

ACKNOWLEDGEMENTS

The authors would like to thank S. Desai, S. Roberts, and K. Johnson from Sandia National Laboratories for their comments and review of this work prior to submission. The authors would also like to thank the anonymous reviewers for their constructive feedback during the review process to improve the overall clarity and quality of the work presented in this paper. This work was supported by the Advanced Engineering Materials program. The deep material network capability and computational resources are supported in part by the Center for Integrated Nanotechnologies, an Office of Science user facility operated by the US Department of Energy. This article has been authored by an employee of National Technology & Engineering Solutions of Sandia, LLC, under Contract No. DE-NA0003525 with the US Department of Energy (DOE). The employee owns all rights, titles, and interests in and to the article and is solely responsible for its contents. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this article or allow others to do so, for United States Government purposes. The DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan https://www.energy.gov/downloads/doe-public-access-plan.

AUTHOR CONTRIBUTIONS

D.S. and R.D. proposed the research idea and designed the overall study. D.S. and R.A. implemented the DMN. D.S. and R.A.L. generated and curated the training/reference FFT data. D.S. performed the numerical analysis of the results. R.D. and R.A.L. provided overall guidance. All the authors analyzed the data, discussed the results, and contributed to writing and reviewing the manuscript.

COMPETING INTERESTS

The authors declare no competing interests.

ADDITIONAL INFORMATION

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41524-023-01085-6.

Correspondence and requests for materials should be addressed to Rémi Dingreville.

Reprints and permission information is available at http://www.nature.com/reprints

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.