Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 371 (2020) 113299
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Geometric deep learning for computational mechanics Part I: anisotropic hyperelasticity

Nikolaos N. Vlassis[a], Ran Ma[a], WaiChing Sun[a,*]

[a] *Department of Civil Engineering and Engineering Mechanics, Columbia University, 610 SW Mudd, Mail Code: 4709, New York, NY 10027, United States of America*

Received 3 January 2020; received in revised form 19 May 2020; accepted 19 July 2020
Available online xxxx

Abstract

We present a machine learning approach that integrates geometric deep learning and Sobolev training to generate a family of finite strain anisotropic hyperelastic models that predict the homogenized responses of polycrystals previously unseen during the training. While hand-crafted hyperelasticity models often incorporate homogenized measures of microstructural attributes, such as the porosity or the averaged orientation of constituents, these measures may not adequately represent the topological structures of the attributes. We fill this knowledge gap by introducing the concept of the weighted graph as a new high-dimensional descriptor that represents topological information, such as the connectivity of anisotropic grains in an assemble. By leveraging a graph convolutional deep neural network in a hybrid machine learning architecture previously used in Frankel et al. (2019), the artificial intelligence extracts low-dimensional features from the weighted graphs and subsequently learns the influence of these low-dimensional features on the resultant stored elastic energy functionals. To ensure smoothness and prevent unintentionally generating a non-convex stored energy functional, we adopt the Sobolev training method for neural networks such that a stress measure is obtained implicitly by taking directional derivatives of the trained energy functional. Results from numerical experiments suggest that Sobolev training is capable of generating a hyperelastic energy functional that predicts both the elastic energy and stress measures more accurately than the classical training that minimizes L_2 norms. Verification exercises against unseen benchmark FFT simulations and phase-field fracture simulations that employ the geometric learning generated elastic energy functional are conducted to demonstrate the quality of the predictions.

Keywords: Geometric machine learning; Graph; Polycrystals; Microstructures; Anisotropic energy functional; Phase-field fracture

1. Introduction

Conventional constitutive modeling efforts often rely on human interpretations of geometric descriptors to incorporate microstructural information into predictions. These descriptors, such as the volume fraction of void/constituents, dislocation density, twinning, degradation function, slip system, orientation, and shape factors are often incorporated as state variables in a system of ordinary differential equations that leads to the constitutive responses at a material point. Classical examples include the family of Gurson models in which the volume fraction

* Corresponding author.
E-mail address: wsun@columbia.edu (W. Sun).

https://doi.org/10.1016/j.cma.2020.113299

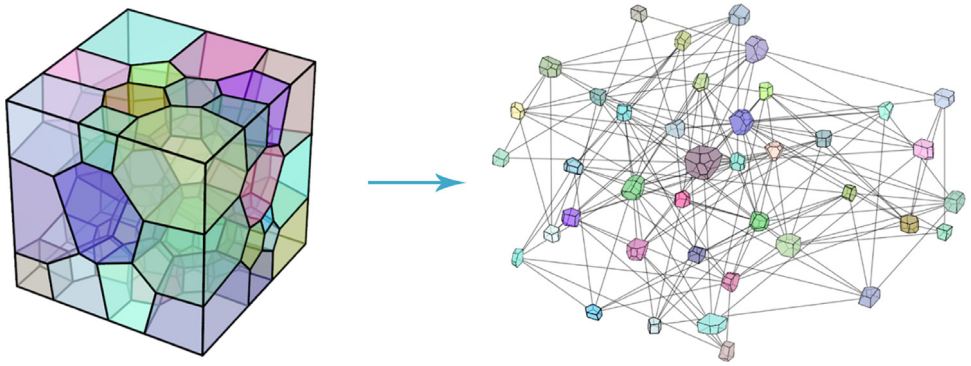


Fig. 1. Polycrystal interpreted as a weighted connectivity graph. The graph is undirected and weighted at the nodes.

of voids is related to ductile fracture [1–6], critical state plasticity in which porosity and over-consolidation ratio dictates the plastic dilatancy and hardening law [7–12] and crystal plasticity where the activation of slip systems leads to plastic deformation [13–15]. In these cases, a specific subset of descriptors is often incorporated manually such that the most crucial deformation mechanisms for the stress–strain relationships are described mathematically.

While this approach has achieved a level of success, especially for isotropic materials, materials of complex microstructures often require more complex geometric and topological descriptors to sufficiently describe the geometrical features [16–19]. The human interpretation limits the complexity of the state variables and may lead to lost opportunity of utilizing all the available information for the microstructure, which could in turn reduce the prediction quality. A data-driven approach should be considered to discover constitutive law mechanisms when human interpretation capabilities become restrictive [20–25]. In this work, we consider the general form of a strain energy functional that reads,

$$\psi = \psi(\boldsymbol{F}, \mathbb{G}) \ , \quad \boldsymbol{P} = \frac{\partial \psi}{\partial \boldsymbol{F}}, \tag{1}$$

where $\mathbb{G}$ is a graph that stores the non-Euclidean data of the microstructures (e.g. crystal connectivity, grain connectivity). Specifically, we attempt to train a neural network approximator of the anisotropic stored elastic energy functional across different polycrystals with the sole extra input to describe the anisotropy being the weighted crystal connectivity graph (see Fig. 1).

It can be difficult to directly incorporate either Euclidean or non-Euclidean data to a hand-crafted constitutive model. There have been attempts to infer information directly from scanned microstructural images using neural networks that utilize a convolutional layer architecture (CNN) [26]. The endeavor to distill physically meaningful and interpretable features from scanned microstructural images stored in a Euclidean grid can be a complex and sometimes futile process. While recent advancements in convolutional neural networks have provided an effective means to extract features that lead to extraordinary superhuman performance for image classification tasks [27], similar success has not been recorded for mechanics predictions. Image-related problems, such as camera noise, saturation, image compression as well as ring artifacts, which often occur in micro-CT images, may lead to issues in the deconvolution operators and, in some cases, may constitute an obstacle in acquiring useful and interpretable features from the image dataset [28]. In some cases, over-fitting and under-fitting can both render the trained CNN extremely vulnerable to adversarial attacks and hence not suitable for high-risk, high-regret applications.

As demonstrated in previous works [29,30], using images directly as an additional input to the polycrystal energy functional approximator may be contingent to the quality and size of the training pool. A large number of images, possibly in three dimensions, and in high enough resolution would be necessary to represent the latent features that will aid the approximator to distinguish successfully between different polycrystals. Using data in a Euclidean grid is an esoteric process that is dependent on empirical evidence that the current training sample holds adequate information to infer features useful in formulating a constitutive law. However, gathering that evidence can be a laborious process as it requires numerous trial and error runs and is weighed down by the heavy computational costs of performing filtering on Euclidean data (e.g. on high resolution 3D image voxels).

Graph representation of the data structures can provide a momentous head-start to overcome this very impediment. An example is the connectivity graph used in granular mechanics community where the formations and evolution of force chains are linked to macroscopic phenomena, such as shear band formation and failures [18,31–35]. The distinct advantage of the graph representation of data is the relatively high interpretability of the data structures [18,34]. This graph representation is not only helpful for understanding the topology of interrelated entities in a network but also provides a convenient means to create universal and interpretable features via graph convolutional neural networks [36,37].

At the same time, by concisely selecting appropriate graph weights, one may incorporate only the essential information of micro-structural data critical for mechanics predictions which could prove to be more interpretable, flexible, economical, and efficient than incorporating feature spaces inferred from 3D voxel images. Furthermore, since one may easily use rotational and transitional invariant data as weights, the graph approach is also advantageous for predicting constitutive responses that require frame indifference.

Currently, machine learning applications often employ two families of algorithms to take graphs as inputs, i.e., representation learning algorithms and graph neural networks. The former usually refers to unsupervised methods that convert graph data structures into formats or features that are easily comprehensible by machine learning algorithms [38]. The latter refers to neural network algorithms that accept graphs as inputs with layer formulations that can operate directly on graph structures [39]. Representation learning on graphs shares concepts with the rather popular embedding techniques on text and speech recognition [40] to encode the input in a vector format that can be utilized by common regression and classification algorithms. There have been multiple studies on encoding graph structures, spanning from the level of nodes [41] up to the level of entire graphs [42,43]. Graph embedding algorithms, like DeepWalk [42], utilize techniques such as random walks to "read" sequences of neighboring nodes resembling reading word sequences in a sentence and encode those graph data in an unsupervised fashion.

While these algorithms have been proven to be rather powerful and demonstrate competitive results in tasks like classification problems, they do come with disadvantages that can be limiting for use in engineering problems. Graph representation algorithms work very well on encoding the training dataset. However, they could be difficult to generalize and cannot accommodate dynamic data structures. This can be proven problematic for mechanics problems, where we expect a model to be as generalized as much as possible in terms of material structure variations (e.g. polycrystals, granular assemblies). Furthermore, representation learning algorithms can be difficult to combine with another neural network architecture for a supervised learning task in a sequential manner. In particular, when the representation learning is performed separately and independently from the supervised learning task that generates the energy functional approximation, there is no guarantee that the clustering or classifications obtained from the representative learning are physically meaningful. Hence, the representation learning may not be capable of generating features that facilitate the energy functional prediction task in a completely unsupervised setting.

For the above reasons, we have opted for a hybrid neural network architecture that combines an unsupervised graph convolutional neural network with a multilayer perceptron to perform the regression task of predicting an energy functional. Both branches of our suggested hybrid architecture learn simultaneously from the same back-propagation process with a common loss function tailored to the approximated function. The graph encoder part – borrowing its name from the popular autoencoder architecture [44,45] – learns and adjusts its weights to encode input graphs in a manner that serves the approximation task at hand. Thus, it does eliminate the obstacle of trying to coordinate the asynchronous steps of graph embedding and approximator training by parallel fitting both the graph encoder and the energy functional approximator with a common training goal (loss function).

As for notations and symbols in this current work, bold-faced letters denote tensors (including vectors which are rank-one tensors); the symbol '$\cdot$' denotes a single contraction of adjacent indices of two tensors (e.g. $\boldsymbol{a}\cdot\boldsymbol{b}=a_ib_i$ or $\boldsymbol{c}\cdot\boldsymbol{d}=c_{ij}d_{jk}$); the symbol ':' denotes a double contraction of adjacent indices of tensor of rank two or higher (e.g. $\boldsymbol{C}:\boldsymbol{\epsilon^e}=C_{ijkl}\epsilon^e_{kl}$); the symbol '$\otimes$' denotes a juxtaposition of two vectors (e.g. $\boldsymbol{a}\otimes\boldsymbol{b}=a_ib_j$) or two symmetric second order tensors (e.g. $(\boldsymbol{\alpha}\otimes\boldsymbol{\beta})_{ijkl}=\alpha_{ij}\beta_{kl}$). Moreover, $(\boldsymbol{\alpha}\oplus\boldsymbol{\beta})_{ijkl}=\alpha_{jl}\beta_{ik}$ and $(\boldsymbol{\alpha}\ominus\boldsymbol{\beta})_{ijkl}=\alpha_{il}\beta_{jk}$. We also define identity tensors $(\boldsymbol{I})_{ij}=\delta_{ij}$, $(\boldsymbol{I}^4)_{ijkl}=\delta_{ik}\delta_{jl}$, and $(\boldsymbol{I}^4_{\text{sym}})_{ijkl}=\frac{1}{2}(\delta_{ik}\delta_{jl}+\delta_{il}\delta_{kj})$, where δ_{ij} is the Kronecker delta. As for sign conventions, unless specified otherwise, we consider the direction of the tensile stress and dilative pressure as positive.

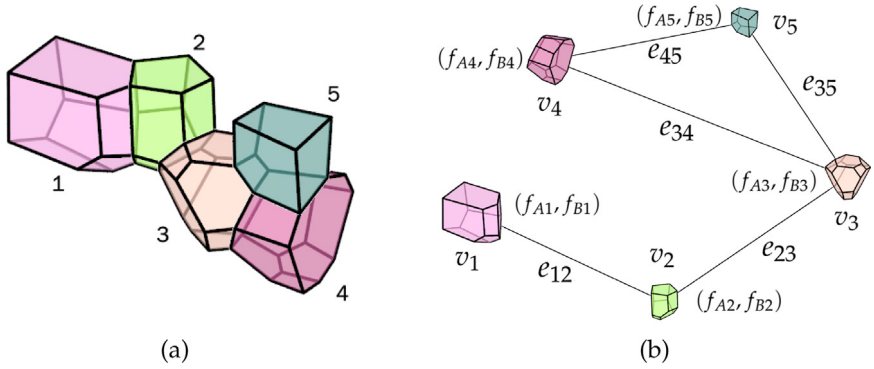


Fig. 2. An illustrative example of a polycrystal (a) represented as an undirected weighted graph (b). If two crystals in the formation share a face, their nodes are also connected in the graph. Each node is weighted by two features f_A and f_B in this illustrative example.

2. Graphs as non-euclidean descriptors for micro-structures

This section provides a detailed account of how to incorporate microstructural data represented by weighted graphs as descriptors for modeling hyperelastic responses of different microstructures. In particular, we describe how topological information of an assemble composed of grains with different properties can be effectively represented as a node-weighted graph (Section 2.1). A brief review of some basic concepts of graph theory is included in Appendix A. An illustrative example of the graph representation for a polycrystal structure in Fig. 2 is included in Appendix B.

2.1. Polycrystals represented as node-weighted undirected graphs

Inferring microstructural data as weighted graphs from field data may require pooling, a down-sampling procedure to convert field data of a specified domain into low-dimensional features that preserve topological information. Examples of applications of pooling inferring include the grain connectivity graph from micro-CT images of assembles [12,46] or realization of micro-structures generated from software packages such as Neper or DREAM.3D [47,48]. In these cases, the node and edge set can be defined in a rather intuitive manner, as the micro-structures are formed by assembles consisting of parts (grains) connected in a specific way represented by the edge set as illustrated in Fig. 2. In this work, we treat each crystal grain as a node or vertex in a graph and create an edge for each in-contact grain pair. Attributes of each grain are represented in a collection of node weights. The features of the edges (such as the contact surface areas, roughness) are neglected to simplify the learning procedures but will be considered in the future. For simplicity, we also assume that the polycrystal contains no voids and the contacts remain intact.

Given a polycrystal microstructure consisting of a finite number of crystal grains N, we define the graph representation used in this work. The shape of the grains is idealized as polyhedrons such that the faces of each grain may be in contact with at most one face of the other grain. As such, an edge is assigned between each in-contact (adjacent) grain pair such that there exist E edges in the graph. The collection of the grains is then represented as a vertex set $\mathbb{V} = \{v_1, \ldots, v_N\}$, and the collection of edges as a edge set $\mathbb{E} \subseteq \mathbb{V} \times \mathbb{V}$. There can only be one unique edge defined between two vertices and the order of the vertices of an edge does not matter — i.e. the pairs are unordered or undirected. As a result, the connectivity of the polycrystal can then be represented by an undirected graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ where $\mathbb{V} = \{v_1, \ldots, v_N\}$ (cf. Definition 1).

Note that $\mathbb{G}$ alone only provides the connectivity information. To predict the elasticity of the polycrystals, more information about the geometrical features and mechanical properties of grains must be extracted in the machine learning process. In our design, these features and properties are stored as weights assigned to each vertex and the purpose of the geometric learning is to find a set of descriptors of lower dimensions than the weighted graph such that they can be directly incorporated into the energy functional calculations. For each vertex v_i in the graph, we define a feature vector $\boldsymbol{f}^i = \{f_1^i, \ldots, f_D^i\}$ where D is the number of node weights that represent the geometrical features (e.g. size, number of faces, aspect ratios) and mechanical properties (e.g. elastic moduli, crystal orientation)

of the grain v_i. In this work, the feature vectors store information about the volume, the crystal orientation (in Euler angles), the total surface area, the number of faces, the numbers of neighbors, as well as other shape descriptors, (e.g. the equivalent diameter) for every crystal grain in the polycrystals. The set of node weights for the entire graph reads

$$\mathbb{F} = \{\boldsymbol{f}^1, \boldsymbol{f}^2, \ldots, \boldsymbol{f}^N\}. \tag{2}$$

Remark 1. Note that including the edge weights in the learning problems is likely to provide a more rich source of data for the learning problems. Information such as the attributes of each contact between each grain pair, including the contact surface areas and the angle of contact — could be used as weights for the edges of the graph. While this current work is solely focused on node-weighted graphs, future work will examine an efficient way to generate energy functional from a node-weighted/edge-weighted graph.

2.2. Adjacency matrix, graph Laplacian and feature matrix

In order to prepare the available data for the geometric learning, it is often more convenient to adopt a matrix representation of a graph. In this work, the geometric learning algorithm used requires the normalized symmetric Laplacian matrix $\boldsymbol{L}^{\text{sym}}$ and the node feature matrix $\boldsymbol{X}$ as inputs (see Appendix A).

First of all, we define the node feature matrix $\boldsymbol{X}$ by simply stacking the feature vectors $\boldsymbol{f}^i$ together. As such, the dimension of the node feature matrix $\boldsymbol{X}$ is $N \times D$. The symmetric normalized Laplacian is obtained from the Laplacian matrix $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ where $\boldsymbol{A}$ and $\boldsymbol{D}$ are the adjacency and degree matrices (cf. Definition 9) and Definition 10) . The adjacency matrix $\boldsymbol{A}$ is a symmetric matrix of dimensions $N \times N$ representing the connectivity of the microstructure. The entries α_{ij} of the matrix $\boldsymbol{A}$ are defined as,

$$\alpha_{ij} = \begin{cases} 1, & v_i \text{ is adjacent to } v_j \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

The degree d_i of a vertex v_i is defined as the total number of adjacent (neighboring) vertices to v_i or, equivalently, the number of crystal grains in contact with the crystal grain v_i. The matrix $\boldsymbol{D}$ is a $N \times N$ diagonal matrix. The entries r_{ij} of the diagonal matrix $\boldsymbol{D}$ are defined as,

$$r_{ij} = \begin{cases} d_i, & i = j \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The symmetric normalized Laplacian matrix $\boldsymbol{L}^{\text{sym}} = \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}$ represents the graph connectivity structure and is one of the inputs for the geometric learning algorithm described in Section 3.2. The matrix $\boldsymbol{L}^{\text{sym}}$ is of $N \times N$ dimensions. The entries l_{ij}^{sym} of the matrix $\boldsymbol{L}^{\text{sym}}$ read,

$$l_{ij}^{\text{sym}} = \begin{cases} 1, & i = j \text{ and } d_i \neq 0 \\ -(d_i d_j)^{-\frac{1}{2}}, & i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Note that while the Laplacian matrix $\boldsymbol{L}$ and the symmetric normalized Laplacian matrix $\boldsymbol{L}^{\text{sym}}$ both represent the connectivity of the grains in the polycrystals, the normalized $\boldsymbol{L}^{\text{sym}}$ is often a more popular choice for spectral-based graph convolutions due to the symmetric and positive-semi-definite properties, as well as properties .

3. Deep learning on graphs

Machine learning often involves algorithms designed to generate functions to represent the available data. Some common applications in machine learning are those of regression and classification. A **regression** algorithm attempts to make predictions of a numerical value provided with input data. A **classification** algorithm attempts to assign a label to an input and place it to one or multiple classes/categories that it belongs to. Classification tasks can be **supervised**, if information for the true labels of the inputs is available during the learning process. Classification tasks can also be **unsupervised**, if the algorithm is not exposed to the true labels of the input during the learning process but attempts to infer labels for the input by learning properties of the input dataset structure. The hybrid geometric learning neural network introduced in this work simultaneously performs an unsupervised

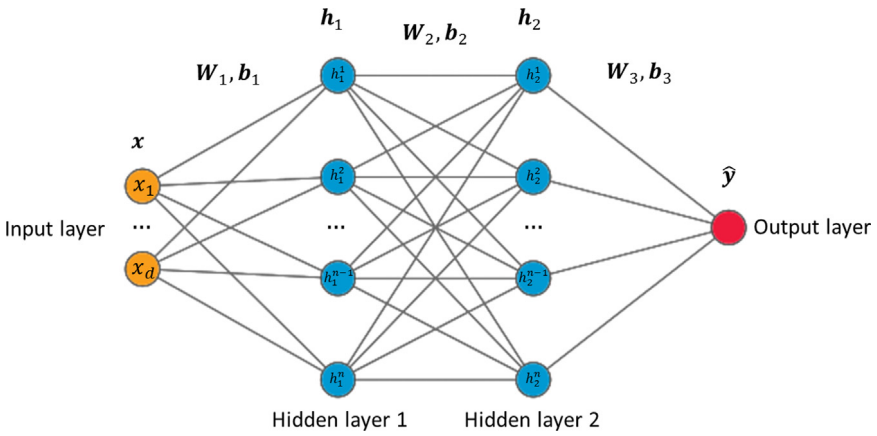


Fig. 3. A two-layer perceptron. The input vector $\boldsymbol{x}$ has d features, each of the two hidden layers h_l has m neurons.

classification of polycrystal graph structures and the regression of an anisotropic elastic energy potential functional. This combination of unsupervised learning classification and supervised learning regression has been first adopted for solid mechanics problems in [30] where convolutional neural network is used to generate features of 3D voxel images to aid predictions of elasto-plastic responses under monotonically increasing strain. The only feature of the grain incorporated in the learning problem is the crystal orientation. In our design, a 3D voxel image is first converted into a lower dimensional weighted graph that contains only connectivity information stored in the symmetric normalized graph Laplacian $\boldsymbol{L}^{\text{sym}}$, while we consider multiple effective properties of each grain stored in the matrix $\boldsymbol{X}$. Then, a geometric learning encoder is trained to provide an even lower dimensional latent representation of the weighted graph to aid the predictions of hyperelastic energy functional.

In this section, we provide a brief description of the supervised and unsupervised components of the hybrid architecture. The supervised learning component conducted via regression with the multilayer perceptron (MLP) is reviewed in Section 3.1. While the graph convolution technique that will carry out the unsupervised classification of the polycrystals is described in Section 3.2. Finally, in Section 3.3, we introduce our hybrid architecture that combines these two architectures to perform their tasks simultaneously.

3.1. Deep learning for supervised regression

The architecture described in this section constitutes the energy functional regression branch of the hybrid architecture described in Section 3.3. The regression task is completed via training an artificial neural network (ANN) with multiple layers. While there are other feasible options, such as support vector regression machines [49] or Gaussian process regression [50], we choose to train a multilayer perceptron (MLP) or often called feed-forward neural network due to the ease of implementations via various existing libraries and the fact that it is a universal function approximator.

The formulation for the two-layer perceptron in Fig. 3, that will also used in this work, is presented below as a series of matrix multiplications:

$$\boldsymbol{z}_1 = \boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1 \tag{6}$$

$$\boldsymbol{h}_1 = \sigma(\boldsymbol{z}_1) \tag{7}$$

$$\boldsymbol{z}_2 = \boldsymbol{h}_1\boldsymbol{W}_2 + \boldsymbol{b}_2 \tag{8}$$

$$\boldsymbol{h}_2 = \sigma(\boldsymbol{z}_2) \tag{9}$$

$$\boldsymbol{z}_{\text{out}} = \boldsymbol{h}_2\boldsymbol{W}_3 + \boldsymbol{b}_3 \tag{10}$$

$$\hat{\boldsymbol{y}} = \sigma_{\text{out}}(\boldsymbol{z}_{\text{out}}). \tag{11}$$

In the above formulation, the input vector $\boldsymbol{x}_l$ contains the features of a sample, the weight matrix $\boldsymbol{W}_l$ contains the weights — parameters of the network, and $\boldsymbol{b}_l$ is the bias vector for every layer. The function σ is the chosen

activation function for the hidden layers. In the current work, the ELU function is used as an activation function for the MLP hidden layers, defined as:

$$\mathrm{ELU}(\alpha) = \begin{cases} e^{\alpha} - 1, & \alpha < 0 \\ \alpha, & \alpha \geq 0. \end{cases} \tag{12}$$

The vector $\boldsymbol{h}_l$ contains the activation function values for every neuron in the hidden layer. The vector $\hat{\boldsymbol{y}}$ is the output vector of the network with linear activation $\sigma_{\text{out}}(\bullet) = (\bullet)$.

Defining $\boldsymbol{y}$ as the true function values corresponding to the inputs $\boldsymbol{x}$, then the MLP architecture could be simplified as an approximator function $\hat{\boldsymbol{y}} = \hat{\boldsymbol{y}}(\boldsymbol{x}|\boldsymbol{W}, \boldsymbol{b})$ of the true function $\boldsymbol{y}$ with inputs $\boldsymbol{x}$ parametrized by $\boldsymbol{W}$ and $\boldsymbol{b}$, such that:

$$\boldsymbol{W}', \boldsymbol{b}' = \underset{\boldsymbol{W},\boldsymbol{b}}{\operatorname{argmin}}\, \ell(\hat{\boldsymbol{y}}(\boldsymbol{x}|\boldsymbol{W}, \boldsymbol{b}), \boldsymbol{y}), \tag{13}$$

where $\boldsymbol{W}'$ and $\boldsymbol{b}'$ are the optimal weights and biases of the neural network that arrive from the optimization — training process such that a defined loss function ℓ is minimized. The loss functions used in this work are discussed in Section 4.

The fully-connected (Dense) layer that is used as the hidden layer for a standard MLP architecture has the following general formulation:

$$\boldsymbol{h}^{(l+1)}_{\text{dense}} = \sigma(\boldsymbol{h}^{(l)}\boldsymbol{W}^{(l)} + \boldsymbol{b}^{(l)}). \tag{14}$$

In the supervised learning branch, the neural network consists of two layers and each layer contains 200 neurons. The number of layers and the number of neurons per layer are hyperparameters. The optimal combination of hyperparameters can be estimated through repeated trial and error or sometimes through Bayesian optimization [51]. To examine if overfitting occurs, we use a k-fold validation to split the training and testing data and measure the differences in performance when the trained neural network is used to make predictions within and outside the training data. A brief review of this issue can be found in [52].

3.2. Graph convolution network for unsupervised classification of polycrystals

Geometric learning refers to the extension of previously established neural network techniques to graph structures and manifold-structured data. Graph Neural Networks (GNN) refer to a specific type of neural networks architectures that operate directly on graph structures. An extensive summary of different graph neural network architectures currently developed can be found in [53]. Graph convolution networks (GCN) [54,55] are variations of graph neural networks that bear similarities with the highly popular convolutional neural network (CNN) algorithms, commonly used in image processing [27,56]. The mutual term convolutional refers to the use of filter parameters that are shared over all locations in the graph similar to image processing. Graph convolution networks are designed to learn a function of features or signals in graphs $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ and they have demonstrated competitive scores at tasks of classification [36,55,57,58].

In this current work, we utilize a GCN layer implementation similar to that introduced in [55]. The implementation is based on the open-source neural network library Keras [59] and the open-source library Spektral [60] on graph neural networks. The GCN layers will be the ones that learn from the polycrystal connectivity graph information. A GCN layer accepts two inputs, a symmetric normalized graph Laplacian matrix $\boldsymbol{L}^{\text{sym}}$ and a node feature matrix $\boldsymbol{X}$ as described in Section 2.1. The matrix $\boldsymbol{L}^{\text{sym}}$ holds information about the graph structure. The matrix $\boldsymbol{X}$ holds information about the features of every node in the graph — every crystal in the polycrystal. In matrix form, the GCN layer has the following structure:

$$\boldsymbol{h}^{(l+1)}_{\text{GCN}} = \sigma(\boldsymbol{L}^{\text{sym}}\boldsymbol{h}^{(l)}\boldsymbol{W}^{(l)} + \boldsymbol{b}^{(l)}). \tag{15}$$

In the above formulation, $\boldsymbol{h}^l$ is the output of a layer l. For $l = 0$, the first GCN layer of the network accepts the graph features as input such that $\boldsymbol{h}^0 = \boldsymbol{X}$. For $l > 1$, $\boldsymbol{h}$ represents a higher dimensional representation of the graph features that are produced from the convolution function, similar to a CNN layer. The function σ is a non-linear activation function. In this work, the GCN layers use the Rectified Linear Unit activation function, defined as $ReLU(\bullet) = \max(0, \bullet)$. The weight matrix $\boldsymbol{W}^l$ and bias vector $\boldsymbol{b}^l$ are the parameters of the layer that will be optimized during training.

The matrix $\boldsymbol{L}^{\text{sym}}$ acts as an operator on the node feature matrix $\boldsymbol{X}$ so that, for every node, the sum of every neighboring node features and the node itself is accounted for. The ith row of the feature matrix $\boldsymbol{h}^{(l)}$ represents the weights for the ith node/crystal in the graph. The output of the multiplication with the ith row of the feature matrix, controlled by the ith row of the $\boldsymbol{L}^{\text{sym}}$ matrix, corresponds to a weighted aggregation output of the features of the ith node and all its neighbors. In order to include the features of the node itself, the matrix $\boldsymbol{L}^{\text{sym}}$ is derived, as defined in Section 2.1 and demonstrated in Appendix B, from the binary adjacency matrix $\hat{\boldsymbol{A}}$ allowing self-loops and the equivalent degree matrix $\boldsymbol{D}$. Using the normalized Laplacian matrix $\boldsymbol{L}^{\text{sym}}$, instead of the adjacency matrix $\hat{\boldsymbol{A}}$, for feature filtering remedies possible numerical and vanishing/exploding gradient issues when using the GCN layer in deep neural networks.

This type of spatial filtering can be of great use in constitutive modeling of microstructures where both the statistics and topology of attributes may both significantly affect the macroscopic outcome. In the case of the polycrystals, for example, the neural network model does not solely learn on the features of every crystal separately. It also learns by aggregating the features of the neighboring crystals in the graph to potentially reveal a behavior that stems from the feature correlation among different nodes. This property deems this filtering function a considerable candidate for learning on spatially heterogeneous material structures.

Remark 2. It is noted that the GCN method can use unweighted graphs as input — in that case the feature matrix is a vector of length N with every component equal to unity, as suggested in [55]. However, due to the significantly less amount of information represented by unweighted graphs, we speculate that the performance of the resultant trained neural network with unweighted graphs is likely to be inferior to that trained on the weighted graph counterparts. The effects of incorporating different combinations of node features on the performance of predictions are examined in the numerical experiments performed in Section 7.

3.3. Hybrid neural network architecture for simultaneous unsupervised classification and regression

The hybrid network architecture employed in this current work is designed to perform two tasks simultaneously, guided by a common objective function. This hybrid design is first applied to mechanics problem by [30] who combine a spatial convolution network with a recurrent neural network to predict constitutive responses. In this work, we adopt the hybrid design where a graph convolutional neural network is combined with a feed-forward neural network to generate elastic stored energy that leads to constitutive responses. While both approaches generate feature vectors as additional inputs for the mechanical predictions, the feature vectors generated from the data stored in voxels (i.e. the Euclidean data) in [30] and the feature vectors generated from the weighted graph (i.e. the non-Euclidean data) are fundamentally different. This is due to the graph convolutional approach requires only grain-scale data where all the feature of the crystal grain is stored as weights at each node, whereas the spatial convolutional approach uses information that is stored at each voxel and, hence, potentially much larger, especially for higher voxel grid resolutions.

The first task is the unsupervised classification of the connectivity graphs of the polycrystals. This is carried through by the first branch of the hybrid architecture that resembles that of a convolutional encoder, commonly used in image classification [27,56] and autoencoders [44,45]. However, the convolutional layers are now following the aforementioned GCN layer formulation. A convolutional encoder passes a complex structure (i.e images, graphs) through a series of filters to generate a higher level representation and encode — compress the information in a structure of lower dimensions (i.e. a vector). It is common practice, for example, in image classification [27], to pass an image through a series of stacked convolutional layers, that increase the feature space dimensionality, and then encode the information in a vector through a multilayer perceptron — a series of stacked fully connected layers. The weights of every layer in the network are optimized using a loss function so that the output vector matches the classification labels of the input image.

A similar concept is employed for the geometric learning encoder branch of the hybrid architecture. This branch accepts as inputs the normalized graph Laplacian and the node feature matrices. The graph convolutional layers read the graph features and increase the dimensionality of the node features. These features are flattened and then fed into fully connected layers that encode the graph information in a feature vector.

The second task performed by the hybrid network is a supervised regression task — the prediction of the energy functional. The architecture of this branch of the network follows that of a simple feed-forward network with

fully connected layers, similar to the one described in Section 3.1. The input of this branch is the encoded feature vector, arriving from the geometric learning encoder branch, concatenated with the second-order right Cauchy–Green deformation tensor $\boldsymbol{C}$ in Voigt vector notation. The output of this branch is the predicted energy functional $\hat{\psi}$. It is noted that in this current work, an elastic energy functional is predicted and the not history-dependent behavior can be adequately mapped with feed-forward architectures. Applications of geometric learning on plastic behavior will be the object of future work and will require recurrent network architectures that can capture the material's behavior history, similar to Wang and Sun [52].

The layer weights of these two branches are updated in tandem with a common back-propagation algorithm and an objective function that rewards the better energy functional and stress field predictions, using a Sobolev training procedure, described in Section 4.

Simultaneously, we implement regularization on the graph encoder branch of the hybrid architecture, in the form of Dropout layers [61]. We have discovered that regularization techniques provide a competent method for combating overfitting issues, addressed later in this work. This work is a first attempt to utilizing geometric learning in material mechanics and the model refinement will be considered when approaching more complex problems in the future (e.g. history-dependent plasticity problems).

4. Sobolev training for hyperelastic energy functional predictions

In principle, forecast engines for elastic constitutive responses are trained by (1) an energy-conjugate pair of stress and strain measures [52,62,63], (2) a power-conjugate pair of stress and strain rates [64] and (3) a pair of strain measure and Helmholtz stored energy [65,66]. While options (1) and (2) can both be simple and easy to train once the proper configuration of the neural networks is determined, one critical drawback is that the resultant model may predict non-convex energy response and exhibit ad-hoc path-dependence [67,68].

An alternative is to introduce supervised learning that takes strain measure as input and output the stored energy functional. This formulation leads to the so-called hyperelastic or Green-elastic material, which postulates the existence of a Helmholtz free-energy function [69]. The concept of learning a free energy function as a means to describe multi-scale materials has been previously explored [70,71]. However, without direct control of the gradient of the energy functional, the predicted stress and elastic tangential operator may not be sufficiently smooth unless the activation functions and the architecture of the neural network are carefully designed. To rectify the drawbacks of these existing options, we leverage the recent work on Sobolev training [73] in which we incorporate both the stored elastic energy functional and the derivatives (i.e. conjugate stress tensor) into the loss function such that the objective of the training is not solely minimizing the errors of the energy predictions but the discrepancy of the stress response as well.

Traditional deep learning regression algorithms aim to train a neural network to approximate a function by minimizing the discrepancy between the predicted values and the benchmark data. However, the metric or norm used to measure discrepancy is often the L_2 norm, which does not regularize the derivative or gradients of the learned function. When combined with the types of activation functions that include a high-frequency basis, the learned function may exhibit spurious oscillations and, hence, be unsuitable for training hyperelastic energy function that requires convexity.

The Sobolev training method that we adopt from Czarnecki et al. [73] is designed to maximize the utilization of data by leveraging the available additional higher order data in the form of higher order constraints in the training objective function. In the Sobolev training, objective functions are constructed for minimizing the H^K Sobolev norms of the corresponding Sobolev space. Recall that a Sobolev space refers to the space of functions equipped with norm comprised of L^p norms of the functions and their derivatives up to a certain order K.

Since it has been shown that neural networks with the ReLU activation function (as well as functions similar to that) can be universal approximators for C^1 functions in a Sobolev space [74], our goal here is to directly predict the elastic energy functional by using the Sobolev norm as loss function to train the hybrid neural network models.

This current work focuses on the prediction of an elastic stored energy functional listed in Eq. (1), thus, for simplicity, the superscript e (denoting elastic behavior) will be omitted for all energy, strain, stress, and stiffness scalar and tensor values herein. In the case of the simple MLP feed-forward network, the network can be seen as an approximator function $\hat{\psi} = \hat{\psi}(\boldsymbol{C}|\boldsymbol{W}, \boldsymbol{b})$ of the true energy functional ψ with input the right Cauchy–Green deformation tensor $\boldsymbol{C}$, parametrized by weights $\boldsymbol{W}$ and biases $\boldsymbol{b}$. In the case of the hybrid neural network architecture, the network can be seen as an approximator function $\hat{\psi} = \hat{\psi}(\boldsymbol{C}, \mathbb{G}|\boldsymbol{W}, \boldsymbol{b})$ of the true energy functional ψ with input

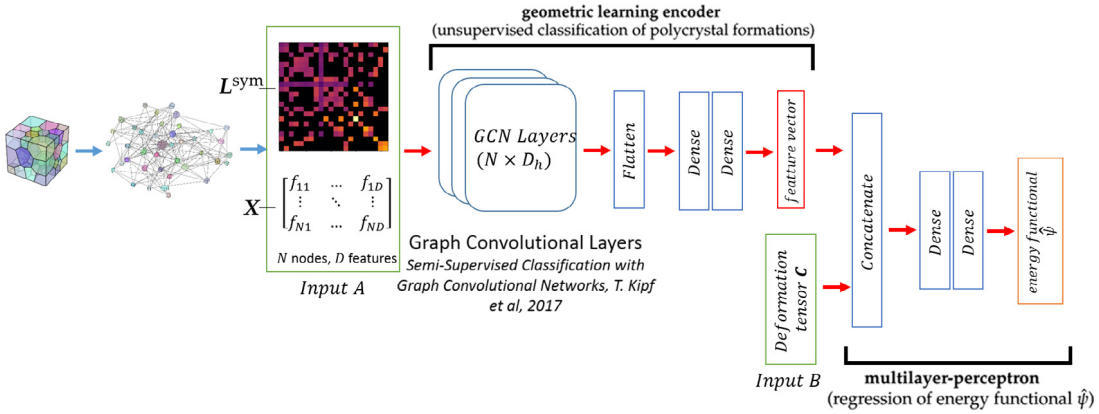


Fig. 4. Hybrid neural network architecture. The network is comprised of two branches — a graph convolutional encoder and a multi-layer perceptron. The first branch accepts the graph structure (normalized Laplacian $\boldsymbol{L}^{\text{sym}}$) and graph weights (feature matrix $\boldsymbol{X}$) (Input A) as inputs and outputs an encoded feature vector. The second branch accepts the concatenated encoded feature vector and the right Cauchy–Green deformation tensor $\boldsymbol{C}$ in Voigt notation (Input B) as inputs and outputs the energy functional $\hat{\psi}$ prediction.

the polycrystal connectivity graph information (as described in Fig. 4) and the tensor $\boldsymbol{C}$, parametrized by weights $\boldsymbol{W}$ and biases $\boldsymbol{b}$. The first training objective in Eq. (16) for the training samples $i \in [1, \ldots, N]$ is modeled after an L_2 norm, constraining only ψ:

$$\boldsymbol{W}', \boldsymbol{b}' = \underset{\boldsymbol{W},\boldsymbol{b}}{\operatorname{argmin}} \left(\frac{1}{N} \sum_{i=1}^{N} \left\| \psi_i - \hat{\psi}_i \right\|_2^2 \right). \tag{16}$$

The second training objective in Eq. (17) for the training samples $i \in [1, \ldots, N]$ is modeled after an H_1 norm, constraining both ψ and its first derivative with respect to $\boldsymbol{C}$ — i.e. one half of the 2nd Piola–Kirchhoff stress tensor $\boldsymbol{S}$:

$$\boldsymbol{W}', \boldsymbol{b}' = \underset{\boldsymbol{W},\boldsymbol{b}}{\operatorname{argmin}} \left(\frac{1}{N} \sum_{i=1}^{N} \left\| \psi_i - \hat{\psi}_i \right\|_2^2 + \left\| \frac{\partial \psi_i}{\partial \boldsymbol{C}_i} - \frac{\partial \hat{\psi}_i}{\partial \boldsymbol{C}_i} \right\|_2^2 \right), \tag{17}$$

where in the above:

$$\boldsymbol{S} = 2 \frac{\partial \psi}{\partial \boldsymbol{C}}. \tag{18}$$

It is noted that higher order objective functions can be constructed as well, such as an H_2 norm constraining the predicted $\hat{\psi}$, stress, and stiffness values. This would be expected to procure even more accurate $\hat{\psi}$ results, smoother stress predictions, and more accurate stiffness predictions. However, since a neural network is a combination of linear functions — the second-order derivative of the ReLU and its adjacent activation functions are zero, it becomes innately difficult to control the second-order derivative during training, thus in this work we mainly focus on the first-order Sobolev method (see Fig. 5). In case it is desired to control the behavior of the stiffness tensor, a first-order Sobolev training scheme can be designed with strain as input and stress as output. The gradient of this approximated relationship would be the stiffness tensor. This experiment would also be meaningful and useful in finite element simulations.

It is noted that, in this current work, the Sobolev training is implemented using the available stress information as the higher order constraint, assuring that the predicted stress tensors are accurate component-wise. In simpler terms, the H_1 norm constrains every single component of the second-order stress tensor. It is expected that this could be handled more efficiently and elegantly by constraining the spectral decomposition of the stress tensor — the principal values and directions. It has been shown in [75] that using loss functions structured to constrain tensorial values in such a manner can be beneficial in mechanics-oriented problems and will be investigated in future work.

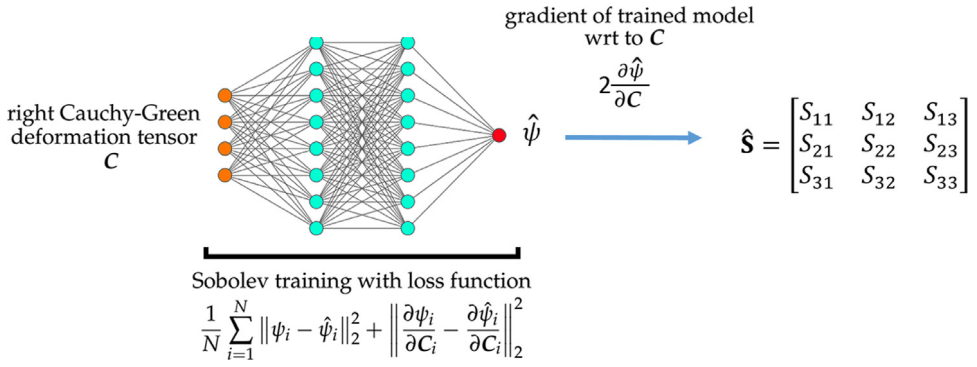


Fig. 5. Schematic of the training procedure of a hyperelastic material surrogate model with the right Cauchy–Green deformation tensor $\boldsymbol{C}$ as input and the energy functional $\hat{\psi}$ as output. A Sobolev trained surrogate model will output smooth $\hat{\psi}$ predictions and the gradient of the model with respect to $\boldsymbol{C}$ will be a valid stress tensor $\hat{\boldsymbol{S}}$.

5. Verification exercises for checking compatibility with physical constraints

While data-driven techniques, such as the neural network architectures discussed in this work, have provided unprecedented efficiency in generating constitutive laws, the consistency of these laws with well-known mechanical theory principles can be rather dubious. Furthermore, Black-box constitutive models generated by training neural network may lack interpretability and therefore verification and validation exercises are essential to ensure that the predictions are compatible with the physics constraints without over-fitting. In this work, we leverage the mechanical knowledge on fundamental properties of hyperelastic constitutive laws to check and – if necessary – enforce the consistency of the approximated material models with said properties. In particular for this work, the generated neural network energy functional models are tested for their objectivity, isotropy (or lack of), and convexity properties. A brief discussion of these desired properties is presented in this section.

5.1. Objectivity

Objectivity requires that the energy and stress response of a deformed elastic body remains unchanged when rigid body motion takes place. The trained models are expected to meet the objectivity condition — i.e. the material response should not depend on the choice of the reference frame. While translation invariance is automatically ensured by describing the material response as a function of the deformation, invariance for rigid body rotations is not necessarily imposed and must be checked. For a given microstructure represented by a graph $\mathbb{G}$, the definition of objectivity for an elastic energy functional ψ formulation is described as follows [20,76]:

$$\psi(\boldsymbol{Q}\boldsymbol{F}, \mathbb{G}) = \psi(\boldsymbol{F}, \mathbb{G}) \text{ for all } \boldsymbol{F} \in GL^{+}(3, \mathbb{R}),\ \boldsymbol{Q} \in SO(3), \tag{19}$$

where $\boldsymbol{Q}$ is a rotation tensor. The above definition can be proven to expand for the equivalent stress and stiffness measures:

$$\boldsymbol{P}(\boldsymbol{Q}\boldsymbol{F}, \mathbb{G}) = \boldsymbol{Q}\boldsymbol{P}(\boldsymbol{F}, \mathbb{G}) \text{ for all } \boldsymbol{F} \in GL^{+}(3, \mathbb{R}),\ \boldsymbol{Q} \in SO(3), \tag{20}$$

$$\mathbb{C}(\boldsymbol{Q}\boldsymbol{F}, \mathbb{G}) = \boldsymbol{Q}\boldsymbol{Q}\mathbb{C}(\boldsymbol{F}, \mathbb{G}) \text{ for all } \boldsymbol{F} \in GL^{+}(3, \mathbb{R}),\ \boldsymbol{Q} \in SO(3). \tag{21}$$

where $\mathbb{C}$ is the Lagrangian tangential elasticity tensor. Thus, a constitutive law is frame-indifferent, if the responses for the energy, the stress, and stiffness predictions are left rotationally invariant. This is automatically satisfied when the response is modeled as an equivalent function of the right Cauchy–Green deformation tensor $\boldsymbol{C}$, since:

$$\boldsymbol{C}^{+} = (\boldsymbol{F}^{+})^{T}\boldsymbol{F}^{+} = \boldsymbol{F}^{T}\boldsymbol{Q}^{T}\boldsymbol{Q}\boldsymbol{F} = \boldsymbol{F}^{T}\boldsymbol{F} \equiv \boldsymbol{C}, \text{ for all } \boldsymbol{F}^{+} = \boldsymbol{Q}\boldsymbol{F}. \tag{22}$$

By training all the models in this work as a function of the right Cauchy–Green deformation tensor $\boldsymbol{C}$, this condition is automatically satisfied and, thus, it will not be further checked.

5.2. Isotropy

For a given microstructure represented by a graph $\mathbb{G}$, the material response described by a constitutive law is expected to be isotropic, if the following is valid:

$$\psi(\boldsymbol{F}\,\boldsymbol{Q},\mathbb{G}) = \psi(\boldsymbol{F},\mathbb{G}) \text{ for all } \boldsymbol{F} \in GL^{+}(3,\mathbb{R}),\ \boldsymbol{Q} \in SO(3). \tag{23}$$

This expands to the stress and stiffness response of the material:

$$\boldsymbol{P}(\boldsymbol{F}\,\boldsymbol{Q},\mathbb{G}) = \boldsymbol{P}(\boldsymbol{F},\mathbb{G})\,\boldsymbol{Q} \text{ for all } \boldsymbol{F} \in GL^{+}(3,\mathbb{R}),\ \boldsymbol{Q} \in SO(3), \tag{24}$$

$$\mathbb{C}(\boldsymbol{F}\,\boldsymbol{Q},\mathbb{G}) = \mathbb{C}(\boldsymbol{F},\mathbb{G})\,\boldsymbol{Q}\,\boldsymbol{Q} \text{ for all } \boldsymbol{F} \in GL^{+}(3,\mathbb{R}),\ \boldsymbol{Q} \in SO(3). \tag{25}$$

Thus, for a material to be isotropic, its response must be right rotationally invariant. In the case that the response is anisotropic, as in the inherently anisotropic material studied in this work, the above should not be valid.

5.3. Convexity

To ensure the thermodynamical consistency of the trained neural network models, the predicted energy functional must be convex. Testing the convexity of a black box data-driven function without an explicitly stated equation is not necessarily a straightforward process. There have been developed certain algorithms to estimate the convexity of black-box functions [77], however, it is outside the scope of this work and will be considered in the future. While convexity would be straight-forward to visually check for a low-dimensional function, this is not necessarily true for a high-dimensional function described by the hybrid models.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex over a compact domain D if for all $x, y \in D$ and all $\lambda \in [0, 1]$, if:

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y). \tag{26}$$

For a twice differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ over a compact domain D, the definition of convexity can be proven to be equivalent with the following statement:

$$f(y) \geq f(x) + \nabla f(x)^T (y - x), \quad \text{for all} \quad x, y \in D. \tag{27}$$

The above can be interpreted as the first-order Taylor expansion at any point of the domain being a global under-estimator of the function f. In terms of the approximated black-box function $\hat{\psi}(\boldsymbol{C},\mathbb{G})$ used in the current work, the inequality (27) can be rewritten as:

$$\hat{\psi}(\boldsymbol{C}_\alpha,\mathbb{G}) \geq \hat{\psi}(\boldsymbol{C}_\beta,\mathbb{G}) + \frac{\partial \hat{\psi}}{\partial \boldsymbol{C}}(\boldsymbol{C}_\beta,\mathbb{G}) : (\boldsymbol{C}_\alpha - \boldsymbol{C}_\beta), \quad \text{for all} \quad \boldsymbol{C}_\alpha, \boldsymbol{C}_\beta \in D. \tag{28}$$

The above constitutes a necessary condition for the approximated energy functional for a specific polycrystal (represented by the connectivity graph $\mathbb{G}$) to be convex, if it is valid for any pair of right Cauchy deformation tensors $\boldsymbol{C}_\alpha$ and $\boldsymbol{C}_\beta$ in a compact domain D. This check is shown to be satisfied in Section 7.3.4.

Remark 3. The trained neural network models in this work will be shown in Section 7 to satisfy the checks and necessary conditions for being consistent with the expected objectivity, anisotropy, and convexity principles. However, in the case where one or more of these properties appears to be absent, it is noted that it can be enforced during the optimization procedure by modifying the loss function. Additional weighted penalty terms could be added to the loss function to promote consistency to required mechanical principles. For example, in the case of objectivity, the additional training objective, parallel to those expressed in Eqs. (16) and (17), could be expressed as:

$$\boldsymbol{W}', \boldsymbol{b}' = \underset{\boldsymbol{W},\boldsymbol{b}}{\operatorname{argmin}} \left(\frac{1}{N} \sum_{i=1}^{N} \lambda \left\| \hat{\psi}(\boldsymbol{Q}\boldsymbol{F},\mathbb{G}|\boldsymbol{W},\boldsymbol{b}) - \hat{\psi}(\boldsymbol{F},\mathbb{G}|\boldsymbol{W},\boldsymbol{b}) \right\|_2^2 \right),\ \boldsymbol{Q} \in SO(3), \tag{29}$$

where λ is a weight variable, chosen between $[0, 1]$, setting the importance of this objective in the now multi-objective loss function, and $\boldsymbol{Q}$ are randomly sampled rigid rotations from the $SO(3)$ group. Constraints of this kind were not deemed necessary in the current paper and will be investigated in future work.

6. FFT offline database generation

This section firstly introduces the fast Fourier transform (FFT) based method for the mesoscale homogenization problem, which was chosen to efficiently provide the database of graph structures and material responses to be used in geometric learning. Following that, the anisotropic Fung hyperelastic model is briefly summarized as the constitutive relation at the basis of the simulations. Finally, the numerical setup is introduced focusing on the numerical discretization, grain structure generation, and initial orientation of the structures in question.

6.1. FFT based method with periodic boundary condition

This section deals with solving the mesoscale homogenization problem using an FFT-based method. Supposing that the mesoscale problem is defined in a 3D periodic domain, where the displacement field is periodic while the surface traction is anti-periodic, the homogenized deformation gradient $\overline{\boldsymbol{F}}$ and first P-K stress $\overline{\boldsymbol{P}}$ can be defined as:

$$\overline{\boldsymbol{F}} = \langle \boldsymbol{F} \rangle, \overline{\boldsymbol{P}} = \langle \boldsymbol{P} \rangle, \tag{30}$$

where $\langle \cdot \rangle$ denotes the volume average operation.

Within a time step, when the average deformation gradient increment $\Delta\overline{\boldsymbol{F}}$ is prescribed, the local stress $\boldsymbol{P}$ within the periodic domain can be computed by solving the Lippmann–Schwinger equation:

$$\boldsymbol{F} + \boldsymbol{\Gamma}^0 * \left(\boldsymbol{P}(\boldsymbol{F}) - \boldsymbol{C}^0 : \boldsymbol{F}\right) = \overline{\boldsymbol{F}}, \tag{31}$$

where $*$ denotes a convolution operation, $\boldsymbol{\Gamma}^0$ is Green's operator, and $\boldsymbol{C}^0$ is the homogeneous stiffness of the reference material. The convolution operation can be conveniently performed in the Fourier domain, so the Lippmann–Schwinger equation is usually solved by the FFT based spectral method [78]. Note that due to the periodicity of the trigonometric basis functions, the displacement field and the strain field are always periodic.

6.2. Anisotropic fung elasticity

An anisotropic elasticity model at the mesoscale level is utilized to generate the homogenized response database for then training the graph-based model in the macroscale. In this section, a generalized Fung elasticity model is utilized as the mesoscale constitutive relation due to its frame-invariance and convenient implementation [79,80].

In the generalized Fung elasticity model, the strain energy density function W is written as:

$$W = \frac{1}{2}c\left[\exp\left(Q\right) - 1\right], \quad Q = \frac{1}{2}\boldsymbol{E} : \boldsymbol{a} : \boldsymbol{E}, \tag{32}$$

where c is a scalar material constant, $\boldsymbol{E}$ is the Green strain tensor, and $\boldsymbol{a}$ is the fourth-order stiffness tensor. The material anisotropy is reflected in the stiffness tensor $\boldsymbol{a}$, which is a function of the spatial orientation and the material symmetry type.

For a material with orthotropic symmetry, the strain energy density can be written in a simpler form as:

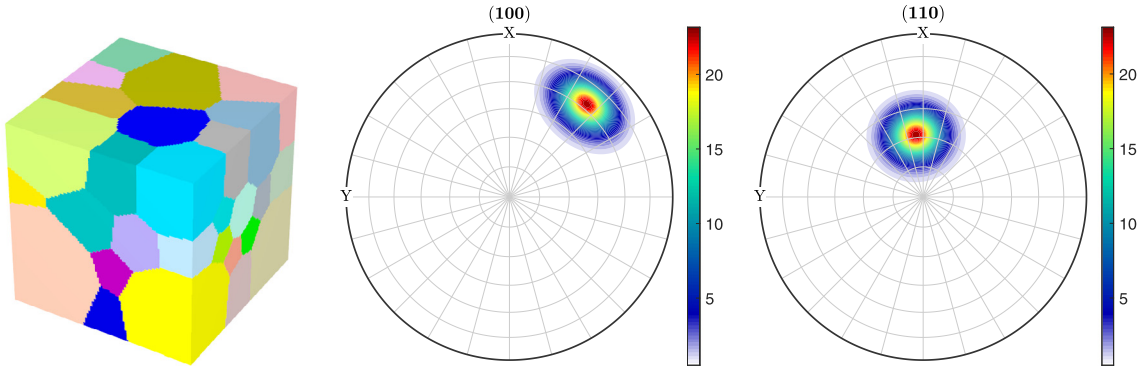
$$Q = c^{-1}\sum_{a=1}^{3}\left[2\mu_a \boldsymbol{A}_a^0 : \boldsymbol{E}^2 + \sum_{b=1}^{3}\lambda_{ab}\left(\boldsymbol{A}_a^0 : \boldsymbol{E}\right)\left(\boldsymbol{A}_b^0 : \boldsymbol{E}\right)\right], \quad \boldsymbol{A}_a^0 = \boldsymbol{a}_a^0 \otimes \boldsymbol{a}_a^0, \tag{33}$$

where μ_a and λ_{ab} are anisotropic Lamé constants, and $\boldsymbol{a}_a^0$ is the unit vector of the orthotropic plane normal, which represents the orientation of the material point in the reference configuration. Note that λ_{ab} is a symmetric second-order tensor, and the material symmetry type becomes cubic symmetry when certain values of $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are adopted.

The elastic constants take the value:

$$c = 2\ (\text{MPa}), \lambda = \begin{bmatrix} 0.6 & 0.7 & 0.6 \\ 0.7 & 1.4 & 0.7 \\ 0.6 & 0.7 & 0.5 \end{bmatrix}\ (\text{MPa}), \mu = \begin{bmatrix} 0.1 \\ 0.7 \\ 0.5 \end{bmatrix}\ (\text{MPa}), \tag{34}$$

and remain constant across all the mesoscale simulations. The only changing variable is the grain structure and the initial orientation of the representative volume element (RVE), which is introduced in the following section.



(a) Sample polycrystal microstructure. **(b)** Sample initial orientation.

Fig. 6. Sample of the randomly generated initial microstructure: (a) Initial RVE with 50 equiaxed grains, which is equally discretized by $49 \times 49 \times 49$ grid points; (b) Pole figure plot of initial orientation distribution function (ODF) combining uniform and unimodal ODF. The Euler angles of the unimodal direction are $(304°, 61°, 211°)$ in Bunge notation, and the half width of the unimodal ODF is 10°. The weight value is 0.50 for uniform ODF and 0.50 for unimodal ODF.

6.3. Numerical aspects of the database generation

The grain structures and initial orientations of the mesoscale simulations are randomly generated in the parameter space to generate the database. The mesoscale RVE is equally divided into $49 \times 49 \times 49$ grid points to maintain a high enough resolution at an acceptable computational cost. The grain structures are generated by the open-source software NEPER [47]. An equiaxed grain structure is randomly generated with 40 to 50 grains. A sample RVE is shown in Fig. 6.

The initial orientations are generated using the open source software MTEX [81]. The orientation distribution function (ODF) is randomly generated by combining uniform orientation and unimodal orientation:

$$f(\boldsymbol{x}; \boldsymbol{g}) = w + (1 - w)\psi\left(\boldsymbol{x}, \boldsymbol{g}\right), \quad \boldsymbol{x} \in SO(3), \tag{35}$$

where $w \in [0, 1]$ is a random weight value, $\boldsymbol{g} \in SO(3)$ is a random modal orientation, and $\psi\left(\boldsymbol{x}, \boldsymbol{g}\right)$ is the von Mises–Fisher distribution function considering cubic symmetry. The half width of the unimodal texture $\psi\left(\boldsymbol{x}, \boldsymbol{g}\right)$ is 10°, and the preferential orientation g of the unimodal texture is also randomly generated. A sample initial ODF is shown in Fig. 6(b).

The average strain is randomly generated from an acceptable strain space, and simulations are performed for each RVE with 200 average strains. Note that the constitutive relation is hyperelastic, so the simulation result is path independent. To avoid any numerical convergence issues, the range of each strain component $(\boldsymbol{F} - \boldsymbol{I})$ is between 0.0 and 0.1 in the RVE coordinate.

7. Numerical experiments

One major advantage of the hybrid [30] or graph-based training [52] is that the resultant neural network is not only suitable to be a surrogate model for one RVE but a family of RVEs with different microstructures. In this section, we present the results of 13 sets of numerical experiments grouped in four subsections to examine and demonstrate the performances of the neural network models we trained. In Section 7.1, we conducted a numerical experiment to examine the neural network trained by the Sobolev method and compare the predictions obtained from the classical loss function that employs L_2 norm. In Section 7.2, we include 4 sets of numerical experiments (a k-fold validation of a Sobolev trained MLP on data for a single polycrystal, a number of input features test for the hybrid architecture, an overfitting check, and a comparison of the k-fold validation results from both the hybrid and the MLP models). In Section 7.3, we include 5 additional verification tests (homogenization experiments, blind predictions for microstructures in the training set, blind predictions for microstructures in the testing set, an isomorphism check, and a model convexity check) to further examine whether the predictions violate any necessary conditions that

Table 1
Summary of the considered model and training algorithm combinations.

Model	Description
$\mathcal{M}_{mlp}^{L2}$	Multilayer perceptron feed-forward architecture. Loss function used is the L_2 norm (Eq. (16))
$\mathcal{M}_{mlp}^{H1}$	Multilayer perceptron feed-forward architecture. Loss function used is the H_1 norm (Eq. (17))
$\mathcal{M}_{hybrid}^{H1}$	Hybrid architecture described in Fig. 4. Loss function used is the H_1 norm (Eq. (17))
$\mathcal{M}_{reg}^{H1}$	Hybrid architecture described in Fig. 4. Loss function used is the H_1 norm (Eq. (17)). The geometric learning branch of the network is regularized against overfitting.

are not explicitly enforced in the objective functions but are crucial for forward predictions. In Section 7.4, we introduced 3 set of tests (dynamic simulations of a single polycrystal, mesh refinement simulations, comparison of crack patterns for different polycrystal inputs) to demonstrate the potential applications of the hyperelastic energy functional predictions for brittle fracture problems using the hybrid architecture. Each of these experiments include multiple calculations that involve both predictions within the calibration range and forward predictions for unseen data previously unused in the training process. The abbreviations we have used for all the model architectures we implemented and tested are summarized in Table 1.

To compare the performance of the training and testing results, the scaled MSE performances of different models are represented using non-parametric, empirical cumulative distribution functions (eCDFs), as in [82,83]. The results are plotted in scaled MSE vs. eCDF curves in a semilogarithmic scale for the training and testing partitions of the dataset. The distance between these curves can be a qualitative metric for the performance of the various models on various datasets — e.g. the distance of the eCDF curves of a model for the training and testing datasets is a qualitative metric of the overfitting phenomenon. For a dataset M with MSE_i sorted in ascending order, the eCDF can be computed as follows:

$$F_M(\text{MSE}) = \begin{cases} 0, & \text{MSE} < \text{MSE}_1, \\ \dfrac{r}{M}, & \text{MSE}_r \leq \text{MSE} < \text{MSE}_{r+1},\ r = 1, \ldots, M-1, \\ 1, & \text{MSE}_M \leq \text{MSE}. \end{cases} \tag{36}$$

To compare all the models in equal terms, the neural network training hyperparameters throughout all the experiments were kept identical wherever it was possible. All the strain and node weight inputs as well as the energy and stress outputs were scaled in the range between [0, 1] during the neural network training. The learning capacity of the models (i.e. layer depth and layer dimensions) for the multilayer perceptrons for $\mathcal{M}_{mlp}^{L2}$ and $\mathcal{M}_{mlp}^{H1}$, as well as the multilayer perceptron branch of the $\mathcal{M}_{hybrid}^{H1}$ and $\mathcal{M}_{reg}^{H1}$ were kept identical. The multilayer perceptron branch in all the networks consists of two Dense layers (200 neurons each) with ELU activation functions. The geometric learning branch consists of two GCN layers (64 filters each with ReLU activation functions) followed by two Dense layers (100 neurons each with ReLU activation functions). The selected encoded feature vector layer was chosen to have 9 neurons. For the $\mathcal{M}_{reg}^{H1}$ model, Dropout layers (dropout rate of 0.2) are defined between every GCN and Dense layer in the geometric learning branch. The optimizer used for the training of the neural networks was Nadam and all the networks were trained for 1000 epochs, utilizing an early stopping algorithm to terminate training when the performance would stop improving. The hyperparameter space of the neural network architecture was deemed rather large to perform a comprehensive parameter search study and the network was tuned through consecutive trial and error iterations. An illustrative example of this trial and error process to tune the number of neurons of the encoded feature vector is demonstrated in Appendix D. In this current work, the values used for the hyperparameters were deemed adequate to provide as accurate results as possible for all methods while maintaining fair comparison terms. The optimization of these hyperparameters to achieve the maximum possible accuracy will be the objective of future work.

Remark 4. Since the energy functional ψ and the stress values are on different scales of magnitude, the prediction errors are demonstrated using a common scaled metric. For all the numerical experiments in this current work, to demonstrate the discrepancy between the predicted values of energy (ψ_{pred}) and the equivalent true energy values (ψ_{true}) as well as between the predicted principal values of the 2nd Piola–Kirchhoff stress tensor ($S_{A,\text{pred}}$) and the

equivalent true principal values ($S_{A,\text{true}}$) for $A = 1, 2, 3$, the following scaled mean squared error (scaled MSE) metrics are defined respectively for a sample of size M:

$$\text{scaled} \quad MSE_{\psi} = \frac{1}{M}\sum_{i=1}^{M}\left[(\overline{\psi}_{\text{true}})_i - (\overline{\psi}_{\text{pred}})_i\right]^2 \quad \text{with} \quad \overline{\psi} := \frac{\psi - \psi_{\min}}{\psi_{\max} - \psi_{\min}}. \tag{37}$$

$$\text{scaled} \quad MSE_{S_A} = \frac{1}{3M}\sum_{i=1}^{M}\sum_{A=1}^{3}\left[(\overline{S}_{A,\text{true}})_i - (\overline{S}_{A,\text{pred}})_i\right]^2 \quad \text{with} \quad \overline{S}_A := \frac{S_A - S_{A,\min}}{S_{A,\max} - S_{A,\min}}, \tag{38}$$

The functions mentioned above scale the values ψ_{pred}, ψ_{true}, $S_{A,\text{pred}}$, and $S_{A,\text{true}}$ to be in the feature range $[0, 1]$. It is noted that the scaling functions $\overline{\psi}$ and $\overline{S}_A$ are defined on the training data set — i.e. the values $\psi_{\min}$, $\psi_{\max}$, $S_{A,\min}$, and $S_{A,\max}$ are derived from the true values of the training data.

Remark 5. When comparing the performance of models in predicting the directions of a second-order stress tensor, we utilize a distance function between two rotation tensors $\boldsymbol{R}_1$, $\boldsymbol{R}_2$ belonging to the Special Orthogonal Group, $SO(3)$. The rotation tensors are constructed by concatenating the orthogonal, normalized eigenvectors of the stress tensors. The Euclidean distance measure ϕ_{Eu}, discussed in detail in [75,84], can be expressed as:

$$\phi_{\text{Eu}} = \sqrt{d\left(\bar{\phi}_1, \bar{\phi}_2\right)^2 + d\left(\bar{\theta}_1, \bar{\theta}_2\right)^2 + d\left(\bar{\psi}_1, \bar{\psi}_2\right)^2}. \tag{39}$$

In the above, $\{\bar{\phi}_i, \bar{\theta}_i, \bar{\psi}_i\} \in \mathbb{E} \subseteq \mathbb{R}^+$ are the set of Euler angles associated with $\boldsymbol{R}_i$, and the Euclidean distance d between two scalar-valued quantities α_1, α_2 is expressed as $d(\alpha_1, \alpha_2) = \min\{|\alpha_1 - \alpha_2|, 2\pi - |\alpha_1 - \alpha_2|\} \in [0, \pi]$. The distance measure ϕ_{Eu} belongs to the range $[0, \pi\sqrt{3}]$ and the results used in the figures in this work are presented normalized in the range $[0, 1]$. For this distance measure, the statement $\phi_{\text{Eu}}(\boldsymbol{R}_1, \boldsymbol{R}_2)$ is equivalent to $\boldsymbol{R}_1 = \boldsymbol{R}_2$.

7.1. Numerical experiment 1: Generating an isotropic hyperelastic energy functional with Sobolev training

In this section, a numerical experiment is presented to demonstrate the benefits of training a neural network on hyperelastic energy functional data in the Sobolev training framework. The experiment was performed on synthetic data generated from a small-strain hyperelastic energy functional designed for the Modified Cam-Clay plasticity model [85–87]. The hyperelastic elastic stored energy functional is described in a strain invariant space (volumetric strain ϵ_v, deviatoric strain ϵ_s). The strain invariants are defined as:

$$\epsilon_v = \text{tr}(\boldsymbol{\epsilon}), \quad \epsilon_s = \sqrt{\frac{2}{3}}\|\boldsymbol{e}\|, \quad \boldsymbol{e} = \boldsymbol{\epsilon} - \frac{1}{3}\epsilon_v \boldsymbol{1}, \tag{40}$$

where $\boldsymbol{\epsilon}$ is the small strain tensor and $\boldsymbol{e}$ the deviatoric part of the small strain tensor. Using the chain rule, the Cauchy stress tensor can be described in the invariant space as follows:

$$\boldsymbol{\sigma} = \frac{\partial \psi}{\partial \epsilon_v}\frac{\partial \epsilon_v}{\partial \boldsymbol{\epsilon}} + \frac{\partial \psi}{\partial \epsilon_s}\frac{\partial \epsilon_s}{\partial \boldsymbol{\epsilon}}. \tag{41}$$

In the above, the mean pressure p and deviatoric (von Mises) stress q can be defined as:

$$p = \frac{\partial \psi}{\partial \epsilon_v} \equiv \frac{1}{3}\text{tr}(\boldsymbol{\sigma}), \quad q = \frac{\partial \psi}{\partial \epsilon_s} \equiv \sqrt{\frac{3}{2}}\|\boldsymbol{s}\|, \tag{42}$$

where $\boldsymbol{s}$ is the deviatoric part of the Cauchy stress tensor. Thus, the Cauchy stress tensor can be expressed by the stress invariants as:

$$\boldsymbol{\sigma} = p\boldsymbol{1} + \sqrt{\frac{2}{3}} q \widehat{\boldsymbol{n}}, \tag{43}$$

$$\text{where } \widehat{\boldsymbol{n}} = \boldsymbol{e}^{\text{e}} / \left\|\boldsymbol{e}^{\text{e}}\right\| = \sqrt{2/3}\, e^{\text{e}} / \epsilon_s^{\text{e}}. \tag{44}$$

The hyperelastic energy functional allows full coupling between the elastic volumetric and deviatoric responses and is described as:

$$\psi(\epsilon_v, \epsilon_s) = -p_0 c_r \exp\left(\frac{\epsilon_{v0} - \epsilon_v}{\kappa}\right) - \frac{3}{2} c_\mu p_0 \exp\left(\frac{\epsilon_{v0} - \epsilon_v}{\kappa}\right)(\epsilon_s)^2, \tag{45}$$

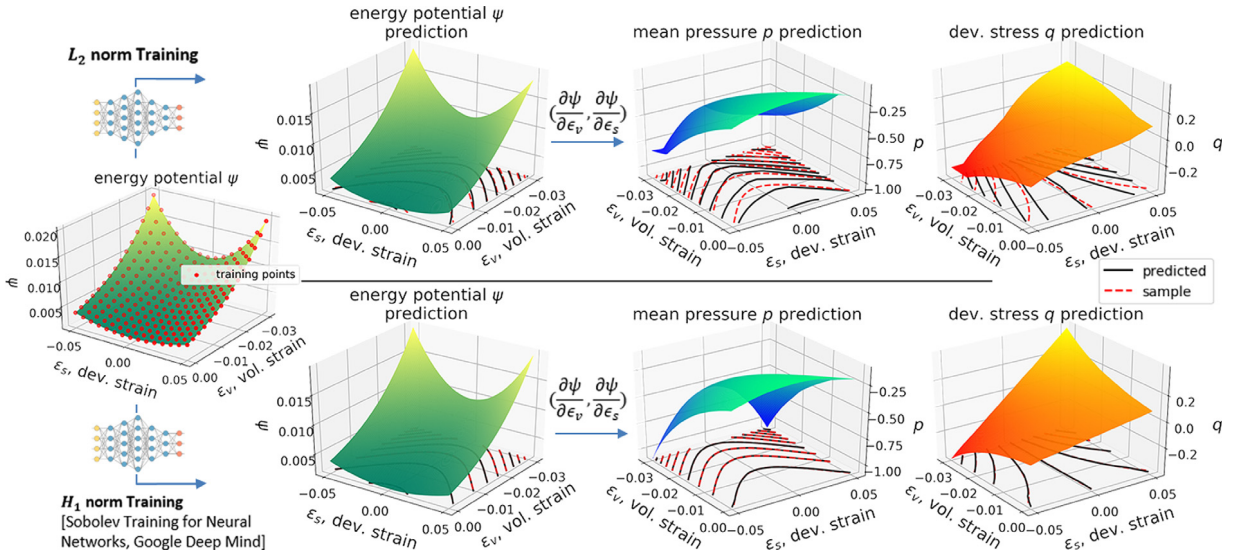


Fig. 7. Comparison of L_2 and H_1 norm training performance for a hyperelastic energy functional used for the Modified Cam-Clay plasticity model [87].

where ϵ_{v0} is the initial volumetric strain, p_0 is the initial mean pressure when $\epsilon_v = \epsilon_{v0}$, $\kappa > 0$ is the elastic compressibility index, and $c_\mu > 0$ is a constant. The hyperelastic energy functional is designed to describe an elastic compression law where the equivalent elastic bulk modulus and the equivalent shear modulus vary linearly with $-p$, while the mean pressure p varies exponentially with the change of the volumetric strain $\Delta\epsilon_v = \epsilon_{v0} - \epsilon_v$. The specifics and the utility of this hyperelastic law is outside the scope of this current work and will be omitted. The numerical parameters of this model were chosen as $\epsilon_{v0} = 0$, $p_0 = -100$ kPa, $c_\mu = 5.4$, and $\kappa = 0.018$. By taking the partial derivatives of the energy functional with respect to the strain invariants, the stress invariants are derived as:

$$p = \frac{\partial \psi}{\partial \epsilon_v} = p_0 \left(1 + \frac{3c_\mu}{2\kappa} (\epsilon_s)^2\right) \exp\left(\frac{\epsilon_{v0} - \epsilon_v}{\kappa}\right), \tag{46}$$

$$q = \frac{\partial \psi}{\partial \epsilon_s} = -3c_\mu p_0 \exp\left(\frac{\epsilon_{v0} - \epsilon_v}{\kappa}\right) \epsilon_s. \tag{47}$$

To compare the performance of the Sobolev training method a two-layer feed-forward neural network is trained on synthetic data generated for the above hyperelastic law. The training data set includes 225 data points, sampled as shown in Fig. 7, 25 of which are randomly selected to be used as a validation set during training. The testing is performed on 1000 data points. The inputs of the neural network are the two strain invariants ϵ_v, ϵ_s and the output is the predicted energy ψ. The network has two hidden Dense layers (100 neurons each) with ELU activation functions and an output Dense layer with a linear activation function. The training experiment is performed with an L_2 norm loss function (constraining only the predicted ψ values) and with an H_1 norm loss function (constraining the predicted ψ, p, and q values).

The results of the two training experiments are shown in Figs. 7 and 8. Both training algorithms seem to be able to capture the energy functional ψ values well with the H_1 trained model demonstrating slightly higher accuracy. However, closer examination in the results shown in Fig. 8 reveals that the neural network trained with a H_1 norm perform better both in predicting the energy functional and the first derivative that leads to the stress invariants p and q. In particular, the neural network trained with the L_2 norm generates a mean pressure and deviatoric stress response that oscillates spuriously with respect to the strain whereas the H_1 counterpart produces results that exhibit no oscillation. Such oscillation is not desirable particularly if the neural network predictions were to be incorporated into an implicit finite element model.

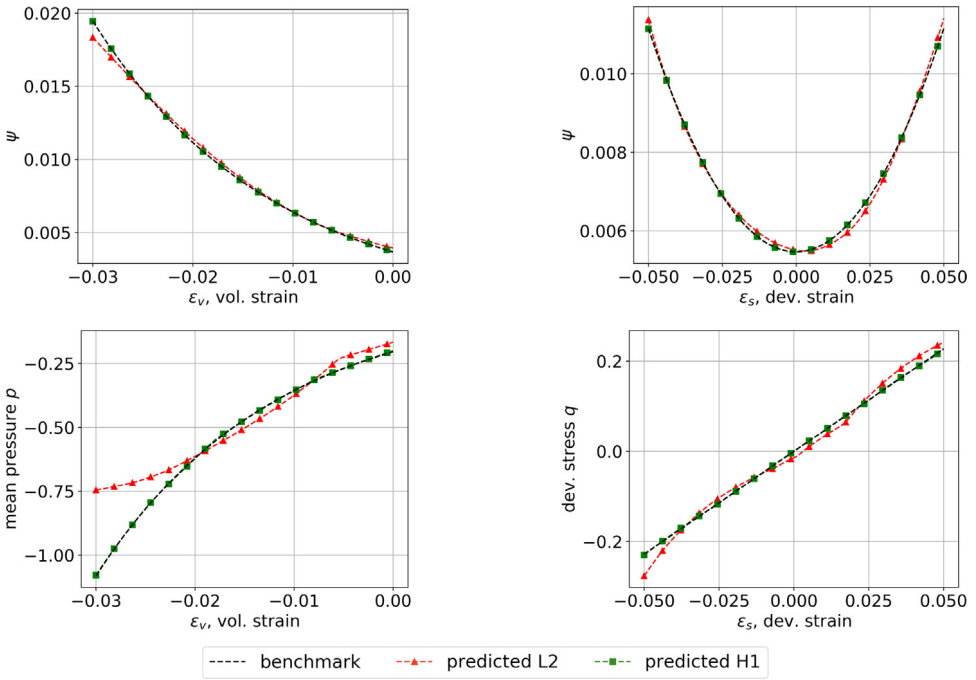


Fig. 8. Comparison of L_2 and H_1 predictions for the energy functional ψ, the stress invariant p, and stress invariant q.

7.2. Numerical experiment 2: Training an anisotropic hyperelastic model for polycrystals with non-Euclidean data

To determine whether the incorporation of graph data improves the accuracy and robustness of the forward prediction, we conduct both the hybrid learning and the classical supervised machine learning. The latter is used as a control experiment. The ability to capture the elastic stored energy functional of a single polycrystal is initially tested on that MLP model. A two-hidden-layer feed-forward neural network is trained and tested on 200 sample points — 200 different, randomly generated deformation tensors with their equivalent elastic stored energy and stress measures for only one of the generated microstructures. A Sobolev trained model as described in Section 4 (model type $\mathcal{M}^{H1}_{mlp}$) was utilized. This architecture will also constitute the multilayer perceptron branch of the hybrid network described previously in Fig. 4. To eliminate as much as possible any objectivity on the dataset of the experiment, the network's capability is tested with a K-fold cross-validation algorithm (cf. [88]). The 200 sample points are separated into 10 different groups — folds of 20 sample points each and, recursively, a fold is selected as a testing set and the rest are selected as a training set for the network.

The K-Fold testing results can be seen in Fig. 9 where the model can predict the data for a single RVE formation adequately, as well as interpolate smoothly between the data points to generate the response surface estimations for the energy and the stress field (Fig. 10). A good performance for both training and testing on a single polycrystal structure was expected as no additional input is necessary, other than the strain tensor. Any additional input – i.e. structural information – would be redundant in the training since it would be constant for the specific RVE.

In this current work, we generalize the learning problem by introducing the polycrystal weighted connectivity graph as the additional input data. This connectivity graph is inferred directly from the micro-structure by assigning each grain in the poly-crystal as a vertex (node) and assigning an edge on each grain contact pair. As discussed in Section 2.1, the nodes of the graphs can have weights carrying information about the crystals they represent in the form of a feature matrix $\boldsymbol{X}$. The available node features in the data set are described in Appendix C. A model of type $\mathcal{M}^{H1}_{hybrid}$ is tested and trained on 150 polycrystals – 100 RVEs in the training set and 50 RVEs in the testing set – with different sets of features to evaluate the effect of the node features to the model's performance. Four combinations of data were tested: a model $\mathcal{M}^{1}_{hybrid}$ with the crystal volume as the only features, a model $\mathcal{M}^{4}_{hybrid}$ with the crystal volume and the crystal Euler angles as features, a model $\mathcal{M}^{8}_{hybrid}$ that utilizes the crystal volumes,

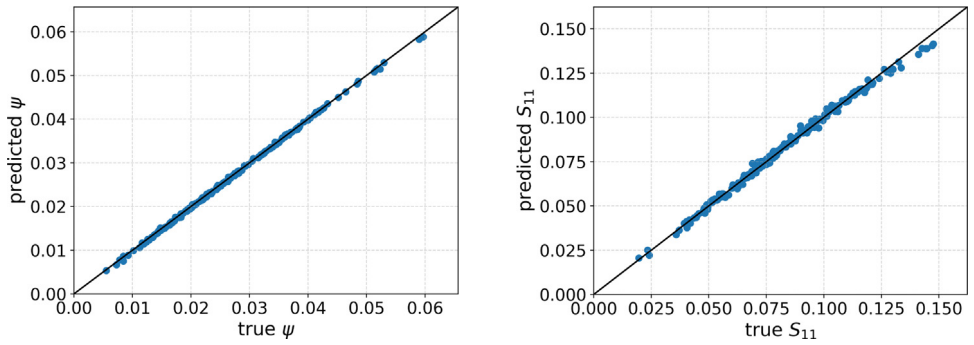


Fig. 9. Correlation plot for true vs. predicted K-fold testing results for the energy functional ψ (left) and the first component of the 2nd Piola–Kirchhoff stress tensor (right) by a surrogate neural network model $\mathcal{M}^{H1}_{mlp}$ trained on data for a single RVE.

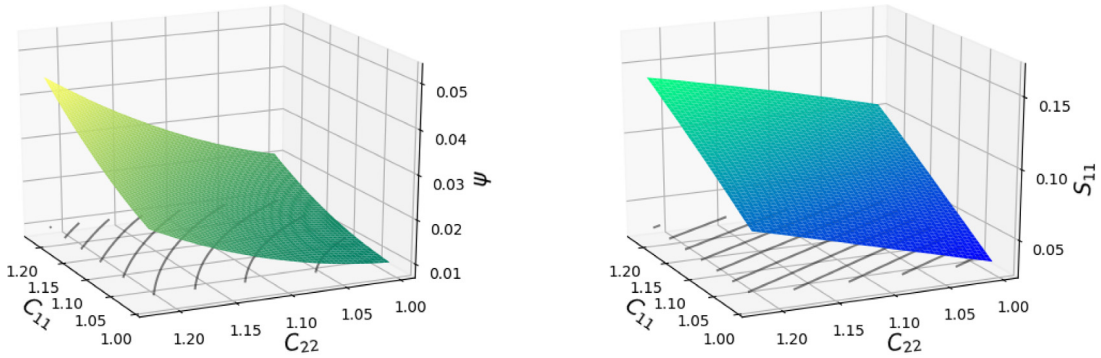


Fig. 10. Estimated ψ energy functional surface (left) and the first component of the 2nd Piola–Kirchhoff stress tensor (right) generated by a surrogate neural network model ($\mathcal{M}^{H1}_{mlp}$) trained on data for a single RVE.

Table 2
Abbreviations of hybrid model names with different number of node weight features.

Model	Node weight features
$\mathcal{M}^{1}_{hybrid}$	Crystal volume
$\mathcal{M}^{4}_{hybrid}$	Crystal volume and three Euler angles
$\mathcal{M}^{8}_{hybrid}$	Crystal volume, three Euler angles, equivalent diameter, number of faces, total area of faces, and number of neighbors
$\mathcal{M}^{11}_{hybrid}$	Crystal volume, three Euler angles, equivalent diameter, number of faces, total area of faces, number of neighbors, and centroid position vector

the Euler angles, the equivalent diameter, the number of faces, the total area of the faces, the number of neighbors, and, finally, a model $\mathcal{M}^{11}_{hybrid}$ that utilizes all the available features. The abbreviations of the model names are also described in Table 2. The results of the training experiment are demonstrated in Fig. 11. Increasing the available node features during training seems to generally increase the model's performance in training and testing. The model that uses the crystal volumes as node features demonstrates the lowest performance. The largest improvement in performance is observed when the crystal Euler angles are included in the feature matrix.

As it was previously mentioned in Section 3.3, the hybrid architecture proposed can be prone to overfitting. To avoid that, we utilize additional Dropout layers in the geometric learning branch of the network as a regularization method during the training. The models representing the hybrid architecture without and with regularization are of the type $\mathcal{M}^{H1}_{hybrid}$ and $\mathcal{M}^{H1}_{reg}$ respectively. To demonstrate that, the two models are tested and trained on 150 polycrystals — 100 RVEs in the training set and 50 RVEs in the testing set. The comparison results are shown in Fig. 12. While $\mathcal{M}^{H1}_{hybrid}$ appears to be prone to overfitting — the training error is lower than the blind prediction error. This issue can be alleviated with regularization techniques that promote the model's robustness in blind

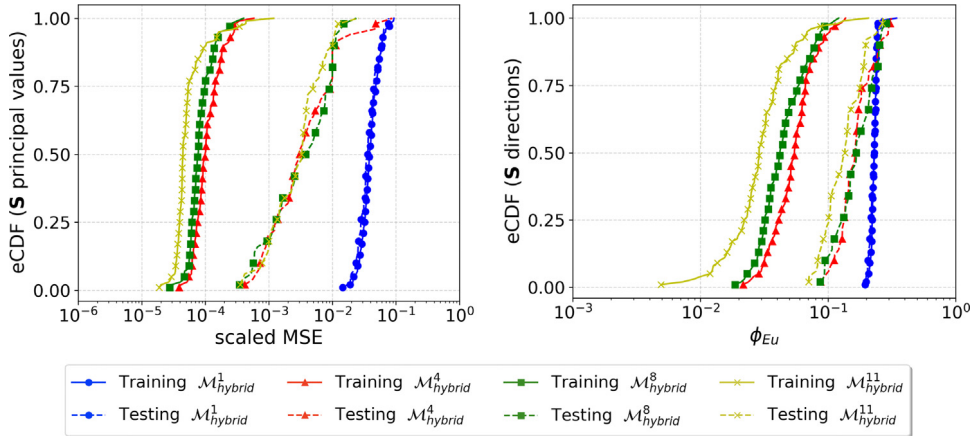


Fig. 11. Comparison of the model's performance for different number of node weight features ($\mathcal{M}^1_{hybrid}$, $\mathcal{M}^4_{hybrid}$, $\mathcal{M}^8_{hybrid}$, and $\mathcal{M}^{11}_{hybrid}$) for the second Piola–Kirchhoff stress $\boldsymbol{S}$ tensor principal values (scaled MSE) and direction predictions (ϕ_{Eu}). The abbreviations of the model names are described in Table 2.

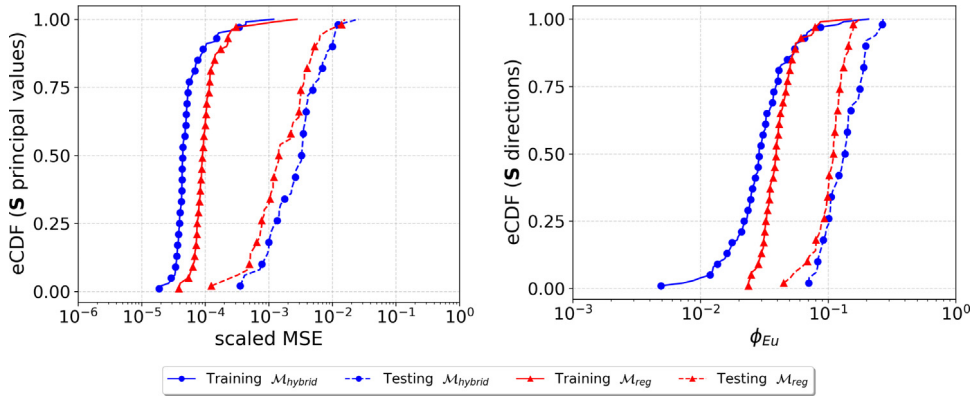


Fig. 12. Scaled mean squared error comparison for the second Piola–Kirchhoff stress $\boldsymbol{S}$ tensor principal value and ϕ_{Eu} error for the $\boldsymbol{S}$ direction predictions for the models $\mathcal{M}^{H1}_{hybrid}$ and $\mathcal{M}^{H1}_{reg}$.

predictions. This can be qualitatively seen on the scaled MSE vs. eCDF plot for the $\mathcal{M}^{H1}_{reg}$ model — the distance between training and testing curves closes for the regularized model. Since the $\mathcal{M}^{H1}_{reg}$ model appears to procure superior results in blind prediction accuracy compared to the not regularized model $\mathcal{M}^{H1}_{hybrid}$, from this point on, we will be working and comparing with the $\mathcal{M}^{H1}_{reg}$ and omitting the $\mathcal{M}^{H1}_{hybrid}$ for simplicity reasons.

The ability of the hybrid architecture proposed in Fig. 4 to leverage the information from a weighted connectivity graph to expand learning over multiple polycrystals in comparison with the classical multilayer perceptron methods is tested in the following experiment. A K-fold validation algorithm is performed on 100 generated polycrystal RVEs. The 100 RVEs are separated into 5 folds of 20 RVEs each. In doing so, every polycrystal RVE will be considered as blind data for the model at least once. The K-fold cross-validation algorithm is repeated for the model architectures and training algorithms $\mathcal{M}^{L2}_{mlp}$, $\mathcal{M}^{H1}_{mlp}$, and $\mathcal{M}^{H1}_{reg}$. The results are presented in Fig. 13 as scaled MSE vs. eCDF curves for the energy functional ψ and second Piola–Kirchhoff stress $\boldsymbol{S}$ tensor principal values and as ϕ_{Eu} vs. eCDF for the principal direction predictions. It can be seen that using the Sobolev training method greatly reduces the blind prediction errors — both the $\mathcal{M}^{L2}_{mlp}$ energy and stress prediction errors are higher than those of the $\mathcal{M}^{H1}_{mlp}$ and $\mathcal{M}^{H1}_{reg}$ models. The $\mathcal{M}^{H1}_{reg}$ model demonstrates superior predictive results than the $\mathcal{M}^{H1}_{mlp}$ model, as it can distinguish between different RVE behaviors.

In addition to the performance measured by this quantitative metric, enabling the weighted graph as an additional input for the hybrid network also provides the opportunity to further generalize the learning process. In Fig. 14,

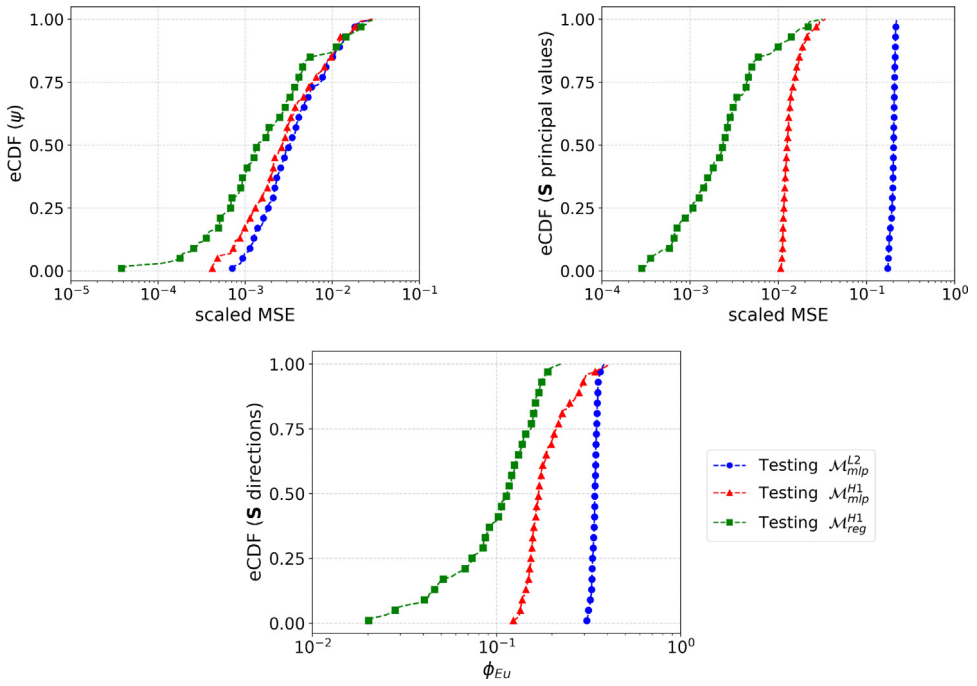


Fig. 13. Scaled MSE vs. eCDF curves for ψ energy functional (top left), scaled MSE vs. eCDF second Piola–Kirchhoff stress $\boldsymbol{S}$ tensor principal values (top right), and ϕ_{Eu} vs. eCDF second Piola–Kirchhoff stress $\boldsymbol{S}$ tensor principal direction predictions (bottom) for the models $\mathcal{M}^{L2}_{mlp}$, $\mathcal{M}^{H1}_{mlp}$, and $\mathcal{M}^{H1}_{reg}$. The models' performance is tested with a K-fold algorithm on a dataset of 100 RVEs — only the blind prediction results are shown.

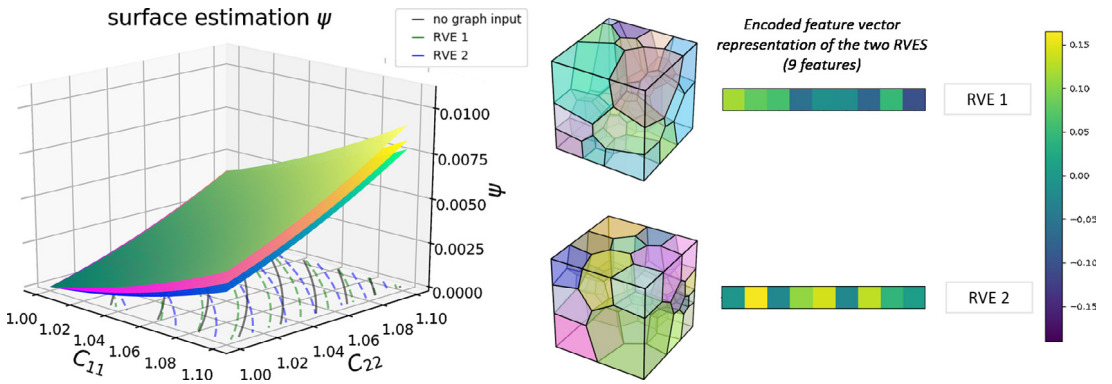


Fig. 14. Without any additional input (other than the strain tensor), the neural network cannot differentiate between these two polycrystals. The two anisotropic behaviors can be distinguished when the weighted connectivity graph is also provided as input. Through the unsupervised encoding branch of the hybrid architecture, each polycrystal is mapped on an encoded feature vector. The feature vector is fed to the multilayer perceptron branch and procures a unique energy prediction.

the energy potential surface estimations are shown for the simple multilayer perceptron and the hybrid architecture for two different polycrystals. Without the graph as input, the network cannot distinguish behaviors, while the hybrid architecture estimates two distinctive elastic stored energy surfaces. The weighted connectivity graph of each polycrystal is encoded in a perceivably different feature vector that aids the downstream multilayer perceptron to identify and interpolate among different behaviors for the RVEs. Furthermore, this hybrid strategy, if trained successfully and carefully validated, is also potentially more efficient than a black-box surrogate model, as the hybrid model does not require a new training process when encountering a new RVE that can be sufficiently described by a weighted graph.

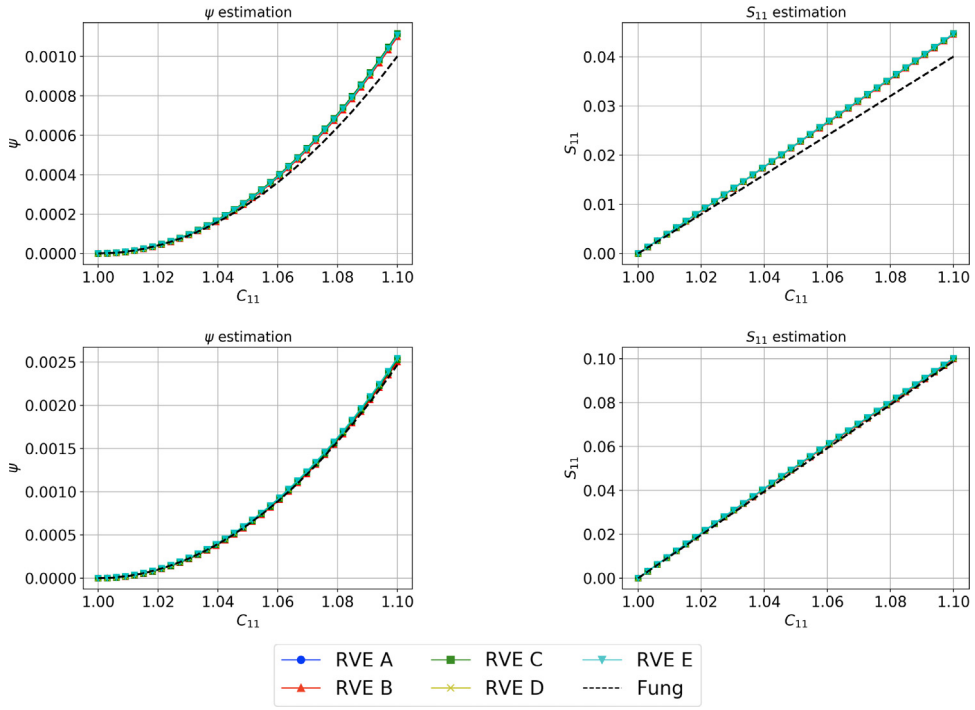


Fig. 15. Estimated ψ and S_{11} responses for 5 unseen RVEs homogenized at Euler angles $(0°, 0°, 0°)$ (top) and at $(45°, 45°, 45°)$ (bottom) of crystal orientations in Bunge notation under uniaxial unconfined tensile loading.

7.3. Numerical experiment 3: Verification tests on unseen data

To ensure that the constitutive response predicted by the trained neural network is consistent with the known mechanics principles, we introduce numerical experiments to verify our ML model and assess the accuracy and robustness of the predictions made by the graph-based model. The predictive capacity of the neural network is tested for blind predictions against homogeneous RVE simulations and new FFT simulations performed on microstructures within and out of the range of the training dataset. The hybrid architecture is also tested on whether it can satisfy the isomorphism condition of the graph inputs and on whether it produces convex energy functionals.

7.3.1. Verification test 1: Responses of unseen homogeneous anisotropic RVEs

In this blind verification, our goal is to check whether the machine learning model predicts the right anisotropic responses of a homogeneous RVE. We use the model trained with the training data described in Appendix C to make a forward prediction on 5 RVEs with all grains of the same crystalline orientation. Since there is no cohesive zone model used in the grain boundary, setting the crystal orientation identical to all the grains essentially makes the RVEs homogeneous.

The ML model then takes the weighted graph that represents the topology of the microstructures as additional input and is used to predict the constitutive responses of these 5 extra microstructures unseen during the training. The results of uniaxial unconfined tensile tests performed on the 5 RVEs of two different orientations $(0°, 0°, 0°)$ and $(45°, 45°, 45°)$ are compared with the benchmark solution as shown in 15. Meanwhile, we also applied pure shear loading on all three pairs of orthogonal planes of these 5 RVES with $(0°, 0°, 0°)$ orientation. The comparison with the benchmark solution is shown in 16.

7.3.2. Verification test 2: Blind test of RVEs with unseen FFT simulations

In this verification test, we test the model's blind prediction capabilities against unseen data generated by FFT simulations. The hybrid architecture model is tested against uniaxial unconfined tension tests and pure shear tests conducted using the FFT solver. The hybrid architecture used in this test was trained on data from 100 RVEs. It

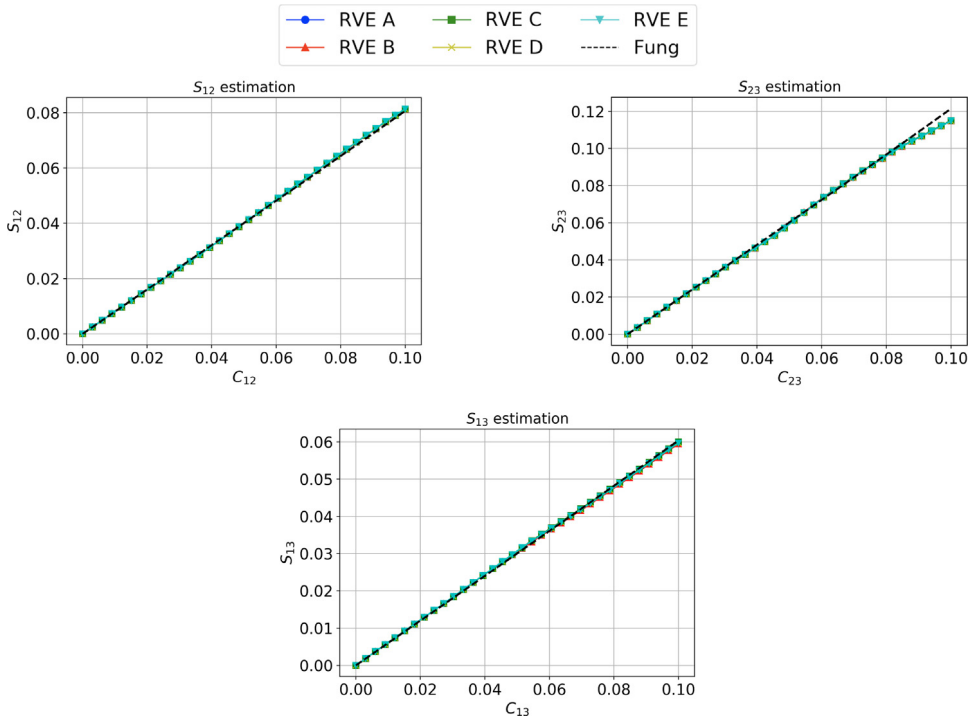


Fig. 16. Estimated S_{12}, S_{23}, and S_{13} responses for 5 unseen RVEs homogenized at Euler angles $(0°, 0°, 0°)$ of crystal orientations in Bunge notation under pure shear loading for 3 directions.

is noted that the model was trained solely on randomly generated deformation gradients and, thus, the strain paths prescribed for these tests are unseen. The results of these tests for three RVEs sampled from the training dataset can be seen in Fig. 17, while the results for three unseen microstructures sampled from the testing dataset can bee seen in Fig. 18.

7.3.3. Verification test 3: Test of graph isomorphism responses

In this test, we check whether the trained geometric learning models procure the same predictions for isomorphic graphs. The definition of graph isomorphism is provided in Appendix A. With the original graph structure of the input known, we can generate any number of isomorphic graphs by applying the same random permutation to the rows and the columns of the original normalized Laplacian matrix of the graph. The same random permutation is applied to the rows of the feature matrix of the input as well. The permuted isomorphic graphs and the equivalent feature matrices carry the same information for the microstructure and the predictions of the hybrid architecture should be consistent under any permutation of the graph input. To test this hypothesis, 10 iterations of permutations were performed on the graph Laplacian matrix and feature matrix of a microstructure to produce isomorphic graph representations of that graph microstructure. These isomorphic graph inputs were then used to make predictions against unseen FFT simulation data. The results of this experiment can be seen in Fig. 19 where the isomorphic graph inputs procure consistent responses.

7.3.4. Verification test 4: Convexity check

To check the convexity for the trained hybrid models, a numerical check was conducted on the trained hybrid architecture models. The models were tested for the check described in Eq. (28). The $\boldsymbol{C}_\alpha$ and $\boldsymbol{C}_\beta$ were chosen to be right Cauchy deformation tensors sampled from the training and testing sets of deformations. The input $\mathbb{G}$ was checked for all the 150 RVEs the hybrid architecture was trained and tested on. For every graph input, the approximated energy functional must be convex. Thus, to verify that for all the poly-crystal formations, the convexity check is repeated for every RVE in the dataset. It is noted that, while these checks describe a necessary

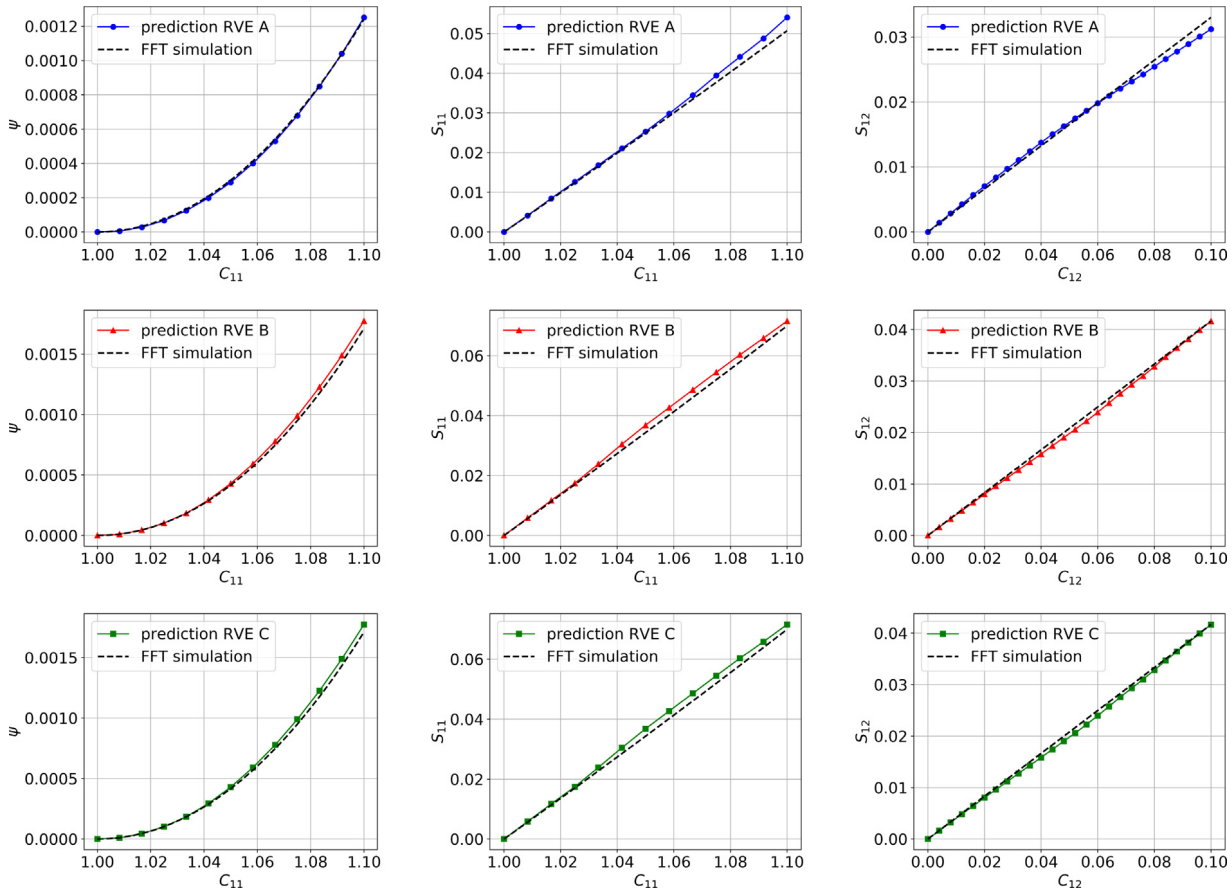


Fig. 17. Comparison of hybrid model predictions with FFT simulation data for 3 RVEs from the training data set. The tests conducted are uniaxial unconfined tension (left and middle columns) and pure shear (right column).

condition for convexity, they do not describe a sufficient condition and more robust methods of checking convexity will be considered in the future. For a specific polycrystal – graph input, the network has six independent variables – deformation tensor $\boldsymbol{C}$ components. To check the convexity, for every RVE in the dataset, deformation tensors $\boldsymbol{C}$ are sampled in a grid and are checked pairwise (approximately 265,000 combinations of points/checks per RVE) and are found to satisfy the inequality (28). In Fig. 20, a sub-sample of 100 convexity checks for three RVEs is demonstrated.

7.4. Numerical experiment 4: Parametric studies on anisotropic responses of polycrystals in phase-field fracture

The anisotropic elastic responses predicted using the hybrid neural network trained by both non-Euclidean descriptors and FFT simulations performed on polycrystals are further examined in the phase-field fracture simulations in which the stored energy functional generated from the hybrid learning model is degraded according to a driving force. In this series of parametric studies, the Kalthoff–Winkler experiment is numerically simulated via a phase-field model in which the elasticity is predicted by the hybrid neural network [89,90]. We adopt the effective stress theory [91] is valid such that the stored energy can be written in terms of the product of a degradation function and the stored elastic energy. The degradation function and the driving force are both pre-defined in this study. The training of an incremental functional for the path-dependent constitutive responses will be considered in the second part of this series of work.

In the first numerical experiment, we conduct a parametric study by varying the orientation of the RVE to analyze how the elastic anisotropy predicted by the graph-dependent energy functional affects the nucleation and propagation

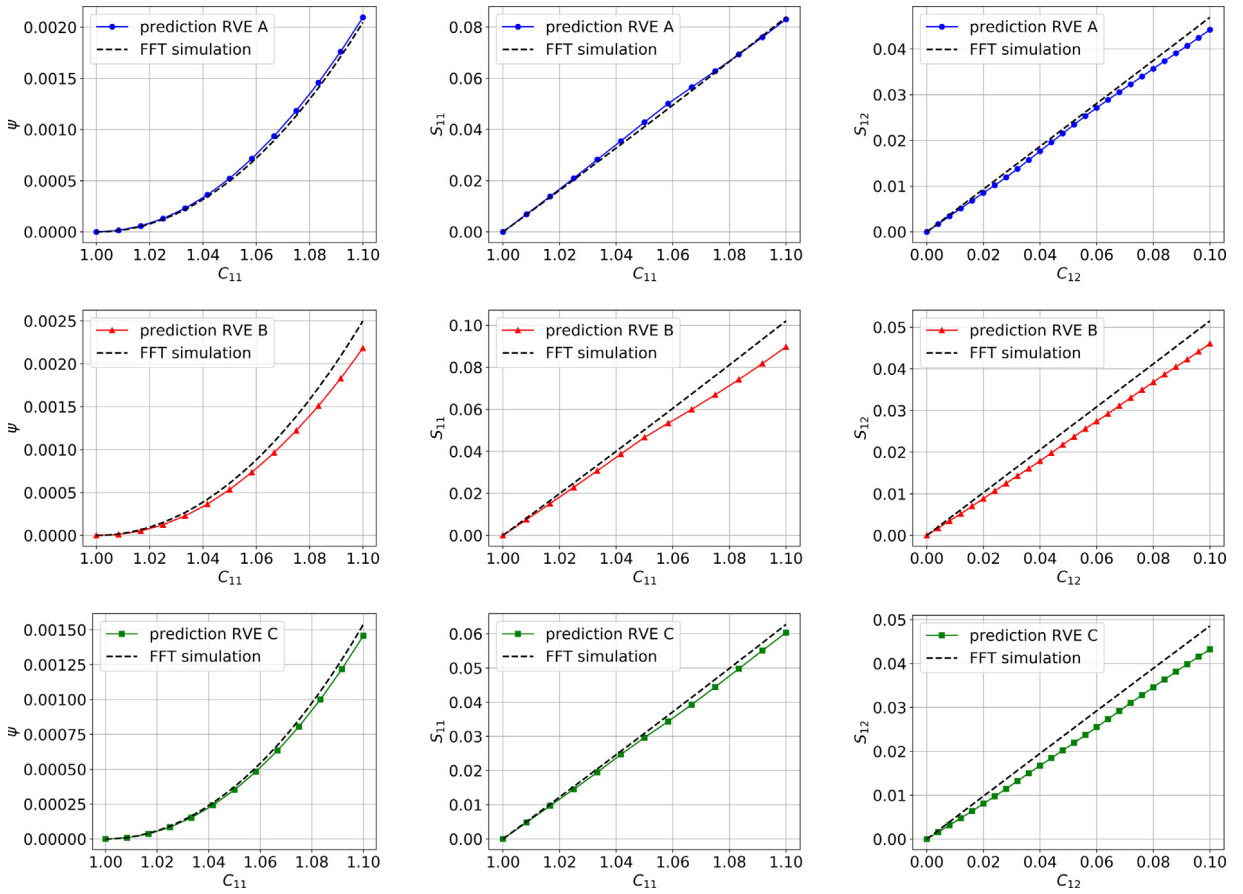


Fig. 18. Comparison of hybrid model predictions with FFT simulation data for 3 RVEs from the testing data set. The tests conducted are uniaxial unconfined tension (left and middle columns) and pure shear (right column).

of cracks. In the second numerical experiment, the hybrid neural network is given new microstructures. Forward predictions of the elasticity of the two new RVEs are made by the hybrid neural network without further calibration. We then compare the crack patterns for the two RVEs and compare the predictions made without the graph input to analyze the impact of the incorporation of non-Euclidean descriptors on the quality of the predictions of crack growths.

While previous work, such as Kochmann et al. [92], has utilized FFT simulations to generate incremental constitutive updates, the efficiency of the FFT–FEM model may highly depend on the complexity of the microstructures and the existence of a sharp gradient of material properties of the RVEs. In this work, the FFT simulations are not performed during the multiscale simulations. Instead, they are used as the training and validation data to generate a ML surrogate model following the treatment in [52] and [35].

For brevity, we omit the detailed description of the phase-field model for brittle fracture. Interested readers please refer to, for instance, Bourdin et al. [93] and Borden et al. [94]. In this work, we adopt the viscous regularized version of phase-field brittle fracture model in Miehe et al. [95] in which the degradation function and the critical energy release rate are pre-defined. The equations solved are the balance of linear momentum and the rate-dependent phase-field governing equation:

$$\nabla^{X} \cdot \boldsymbol{P} + \boldsymbol{B} = \rho \ddot{\boldsymbol{U}}, \tag{48}$$

$$\frac{g_c}{l_0}(d - l_0^2 \nabla^{X} \cdot [\partial_{\nabla d} \gamma]) + \eta \dot{d} = 2(1-d)\mathcal{H}, \tag{49}$$

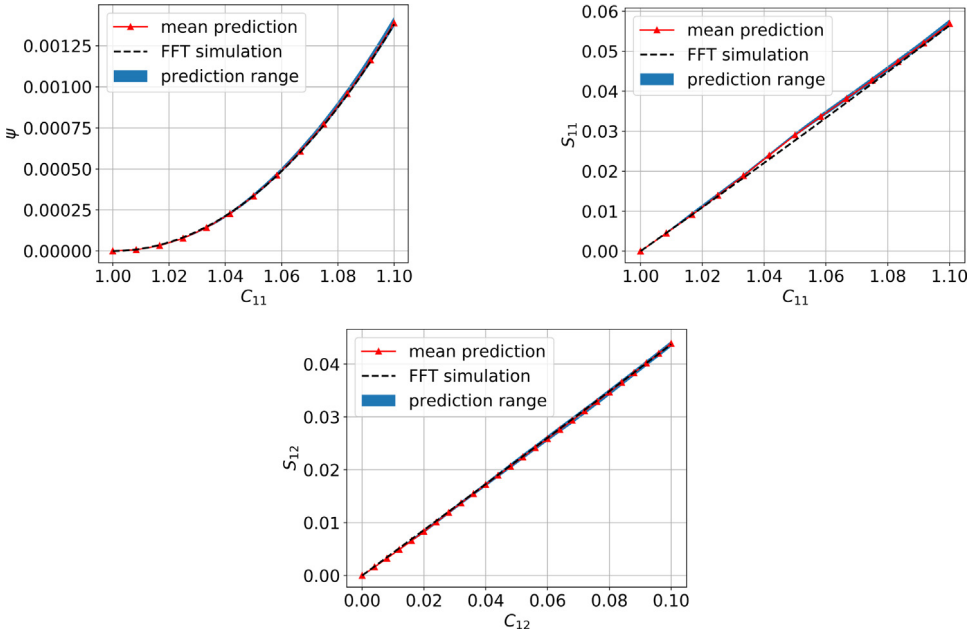


Fig. 19. Hybrid model prediction for isomorphic graph inputs. The mean value and the range of the predictions are compared to the FFT simulation benchmark results for an unconfined uniaxial tension test (top row) and pure shear test (bottom row).

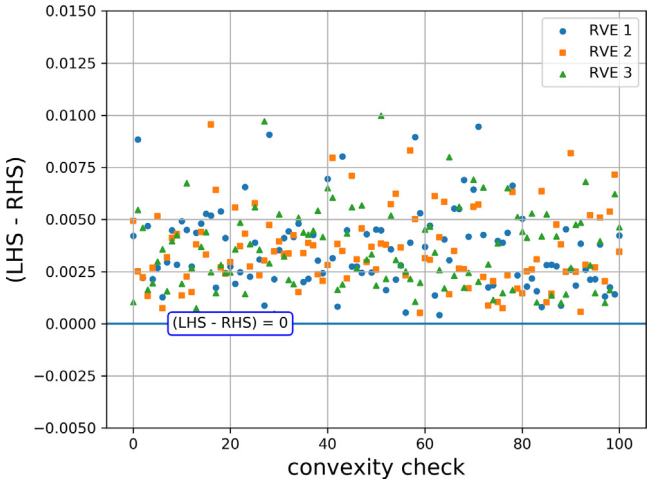


Fig. 20. Approximated energy functional convexity check results for three different polycrystals. Each point represents a convexity check and must be above the [LHS − RHS = 0] line so that the inequality (28) is satisfied.

where γ is the crack density function that represents the diffusive fracture, i.e.,

$$\gamma(d, \nabla d) = \frac{1}{2l_0} d^2 + \frac{l_0}{2} |\nabla d|^2. \tag{50}$$

The problem is solved following a standard staggered time discretization [94] such that the balance of linear momentum and the phase-field governing equations are updated sequentially. In the above Eq. (48), $\boldsymbol{P}$ is the first Piola–Kirchhoff stress tensor, $\boldsymbol{B}$ is the body force and $\ddot{\boldsymbol{U}}$ is the second time derivative of the displacement $\boldsymbol{U}$. In Eq. (49), following, Miehe et al. [95], d refers to the phase-field variable, with $d = 0$ signifying the undamaged and $d = 1$ the fully damaged material. The variable l_0 refers to the length scale parameter used to approximate the sharp crack topology as a diffusive crack profile, such that as $l_0 \to 0$ the sharp crack is recovered. The parameter

g_c is the critical energy release rate from the Griffith crack theory. The parameter η refers to an artificial viscosity term used to regularize the crack propagation by giving it a viscous resistance. The term $\mathcal{H}$ is the force driving the crack propagation and, in order to have an irreversible crack propagation in tension, it is defined as the maximum tensile ("positive") elastic energy that a material point has experienced up to the current time step t_n, formulated as:

$$\mathcal{H}(\boldsymbol{F}_{t_n}, \mathbb{G}) = \max_{t_n \geq t} \psi^+(\boldsymbol{F}_{t_n}, \mathbb{G}). \tag{51}$$

The degradation of the energy due to fracture should take place only under tension and can be linked to that of the undamaged elastic solid as:

$$\psi(\boldsymbol{F}, d, \mathbb{G}) = (g(d) + r)\psi^+(\boldsymbol{F}, \mathbb{G}) + \psi^-(\boldsymbol{F}, \mathbb{G}). \tag{52}$$

The parameter r refers to a residual energy remaining even in the full damaged material and it is set $r \approx 0$ for these experiments. For these numerical experiments, the degradation function that was used was the commonly used quadratic [95]:

$$g(d) = (1-d)^2 \quad \text{with} \quad g(0) = 1 \quad \text{and} \quad g(1) = 0. \tag{53}$$

In order to perform a tensile–compressive split, the deformation gradient is split into a volumetric and an isochoric part. The energy and the stress response of the material should not be degraded under compression. The split of the deformation gradient, following [96], is performed as follows:

$$\boldsymbol{F} = \boldsymbol{F}_{\text{iso}}\boldsymbol{F}_{\text{vol}} = \boldsymbol{F}_{\text{vol}}\boldsymbol{F}_{\text{iso}}, \tag{54}$$

where the volumetric component of $\boldsymbol{F}$ is defined as

$$\boldsymbol{F}_{\text{vol}} = (\det \boldsymbol{F})^{1/3}\boldsymbol{I}, \tag{55}$$

and the volume-preserving isochoric component as

$$\boldsymbol{F}_{\text{iso}} = (\det \boldsymbol{F})^{-1/3}\boldsymbol{F}. \tag{56}$$

The strain energy is, thus, split in a "tensile" and "compressive" part, such that:

$$\psi^+ = \begin{cases} \psi(\boldsymbol{F}, \mathbb{G}) & J \geq 1 \\ \psi(\boldsymbol{F}, \mathbb{G}) - \psi(\boldsymbol{F}_{\text{vol}}, \mathbb{G}) & J < 1, \end{cases} \tag{57}$$

$$\psi^- = \begin{cases} 0 & J \geq 1 \\ \psi(\boldsymbol{F}_{\text{vol}}, \mathbb{G}) & J < 1. \end{cases} \tag{58}$$

where $J = \det(\boldsymbol{F})$. In these examples, the energy values are calculated using the hybrid architecture neural network model, whose derivatives with respect to the strain input will be the stress. Since the model's input is in terms of the right Cauchy–Green deformation tensor, the degraded stress is calculated as:

$$\boldsymbol{P}(\boldsymbol{F}, d, \mathbb{G}) = 2\boldsymbol{F}\left[g(d)\frac{\partial \hat{\psi}^+(\boldsymbol{C}, \mathbb{G})}{\partial \boldsymbol{C}} + \frac{\partial \hat{\psi}^-(\boldsymbol{C}, \mathbb{G})}{\partial \boldsymbol{C}}\right]. \tag{59}$$

The experiment in question studies the crack propagation due to the high velocity impact of a projectile. The geometry and boundary conditions of the domain, as well as the configuration of the pre-existing crack, is shown in Fig. 21. It is noted that, while only half of the domain of the problem is studied in this work, the rest of the domain would not necessarily demonstrate a symmetric response due to the material's anisotropic behavior. In this preliminary study, it was deemed that the half domain would be adequate to illustrate and compare the different anisotropic responses of the model in question. Kalthoff and Winkler [89] and Kalthoff [90] have observed the crack to propagate at 70° for an isotropic material, results that have previously been reproduced with numerical simulations in other studies [72,94,97,98]. The experiment is conducted for two impact velocities ($v_0 = 16.5$ m/s and $v_0 = 33.0$ m/s) to test the crack branching phenomenon expected for higher impact velocities.

The experiment lasts for 80 μs and the prescribed velocity is applied progressively following the scheme below for $t_0 = 1$ μs:

$$v = \begin{cases} \frac{t}{t_0}v_0 & t \leq t_0 \\ v_0 & t > t_0. \end{cases} \tag{60}$$

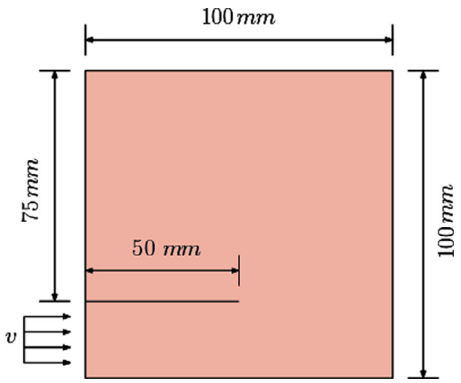


Fig. 21. The geometry and boundary conditions of the domain for the dynamic shear loading experiment. The velocity is prescribed progressively at the bottom left corner of the domain. The mesh is designed to have a pre-existing crack of 50.0 mm.

The domain is meshed uniformly with 20,000 triangular elements and the length scale is chosen to be $l_0 = 1.2 \times 10^{-3}$ m. While this mesh is rather coarse compared to previous studies of the same problem, it was deemed adequate to simulate the problem at hand with acceptable accuracy to qualitatively demonstrate the anisotropic model behavior. The time-step used for the explicit method was chosen to be $\Delta t = 5 \times 10^{-8}$ s to provide stable results. Changing the time step of the explicit method did not appear to affect the phase-field solution, as long as the explicit solver for the momentum equation was stable.

For the first numerical example, the experiment is initially conducted on an isotropic material with parameters commonly used in the literature ($E = 190$ GPa, $\nu = 0.3$, $l_0 = 1.2 \times 10^{-3}$ m) to verify the formulation and compare with the anisotropic results. It can be seen that with the current formulation, the isotropic model can recover the approximately 70° angle previously reported in the experiments and numerical simulations. Following that, the behavior of a single polycrystal was tested. In other words, the graph input of the hybrid architecture material model remained constant for all the simulations. The material model is a trained neural network of type $\mathcal{M}^{H1}_{reg}$ with the graph input set constant. All the neural networks used in this section were trained on a dataset of 100 RVEs with 200 sample points each. The purpose of this experiment is to show that by rotating the highly anisotropic RVE, under the same boundary conditions, different wave propagation and crack nucleation patterns can be observed. This experiment could be paralleled to rotating a transversely isotropic material — different fiber orientations should procure different results under identical boundary conditions. In Fig. 22, it is demonstrated that the neural network material model is indeed anisotropic, showing varying behaviors while rotating the RVE for 0°, 30°, and 60°. The nature of the anisotropy becomes more apparent when the impact velocity is doubled and the crack branching is more prevalent.

Dynamic simulations can be prone to numerical instabilities that may affect the predicted crack propagation patterns [99]. To ensure the crack propagation patterns demonstrated in this experiment are not prone to numerical instabilities depending on the mesh, the simulations were repeated on different meshes and different levels of mesh refinement. The simulation shown in Fig. 22(d) ($v = 33.0$ m/s, $\phi = 0°$) was repeated on a mesh with 11,450 quadrilateral elements, as well as a mesh with 80,000 triangular elements. For the quadrilateral elements and the refined triangular meshes, the simulation time step were chosen to be $\Delta t = 5 \times 10^{-8}$ s and $\Delta t = 2.5 \times 10^{-8}$ s respectively. The simulation shown in Fig. 22(h) ($v = 33.0$ m/s, $\phi = 60°$) was also repeated on a mesh with 80,000 triangular elements to investigate whether the mesh would affect the crack propagation patterns of the RVEs under rotation. The comparison of the crack patterns is demonstrated in Figs. 23 and 24. Neither the selection of element shape nor the level of refinement seem to be greatly affecting the propagated cracks. The main cracks for all the simulations at different levels of refinement appear to be close to identical. There are secondary cracks in Fig. 23(h) as well as Fig. 23(b) appear to be more defined, which is expected due to the higher resolution of the mesh.

For the second numerical experiment, the material response was tested for different polycrystals (model type $\mathcal{M}^{H1}_{reg}$) as well as for a model without any graph inputs (type $\mathcal{M}^{H1}_{mlp}$). The aim of this experiment was to verify that the hybrid architecture and the graph input can capture the anisotropy of the polycrystal material that is originating from the interactions between crystals, as expressed by the connectivity graph. The above experiment was repeated

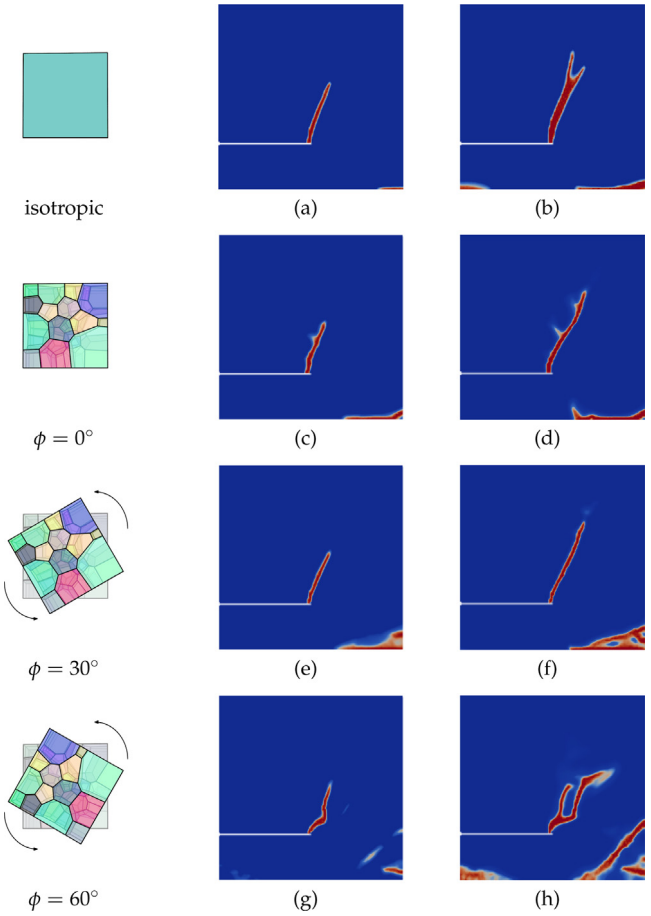


Fig. 22. Crack patterns at 65 μs for the dynamic shear loading experiment for the isotropic material and the anisotropic material for a constant graph, rotated at various angles. The left column shows the experiments for $v = 16.5$ m/s and the right column for $v = 33.0$ m/s.

for different graph inputs and the results are demonstrated in Fig. 25. In the absence of a graph input, while there is crack propagation, the results look noisy and the direction of the propagation is not similar to that of specific RVEs, something that could be potentially attributed to the model being trained on multiple polycrystal behaviors. For the model with the graph input, the difference in behaviors appears to become more apparent in the areas where branching is more prevalent, with the polycrystal affecting the crack branching phenomena. No additional anisotropy measures or crack branching criteria were utilized for these simulations. The sole additional information in the input of the material model would be the weighted connectivity graph.

8. Conclusion

We introduce a machine learning method that incorporates geometric learning to extract low-dimensional descriptors from microstructures represented by weighted graphs and use these descriptors to enhance the supervised learning procedure such that it may generate a family of stored elastic energy functionals for arbitrary microstructures via Sobolev training. By utilizing non-Euclidean data structures, we introduce these weighted graphs as new descriptors for geometric learning such that the hybrid deep learning can produce an energy functional that leverages the rich micro-structural information not describable by the classical Euclidean descriptors, such as porosity and density. To overcome the potential spurious oscillations of the learned functions due to lack of constraints on their derivatives, we adopt the Sobolev training and the resultant hyperelastic energy functional is more accurate and smoother, compared to classical machine learning techniques. This work also laid the foundation for several new potential research directions. For instance, the energy functional approach can be extended to a

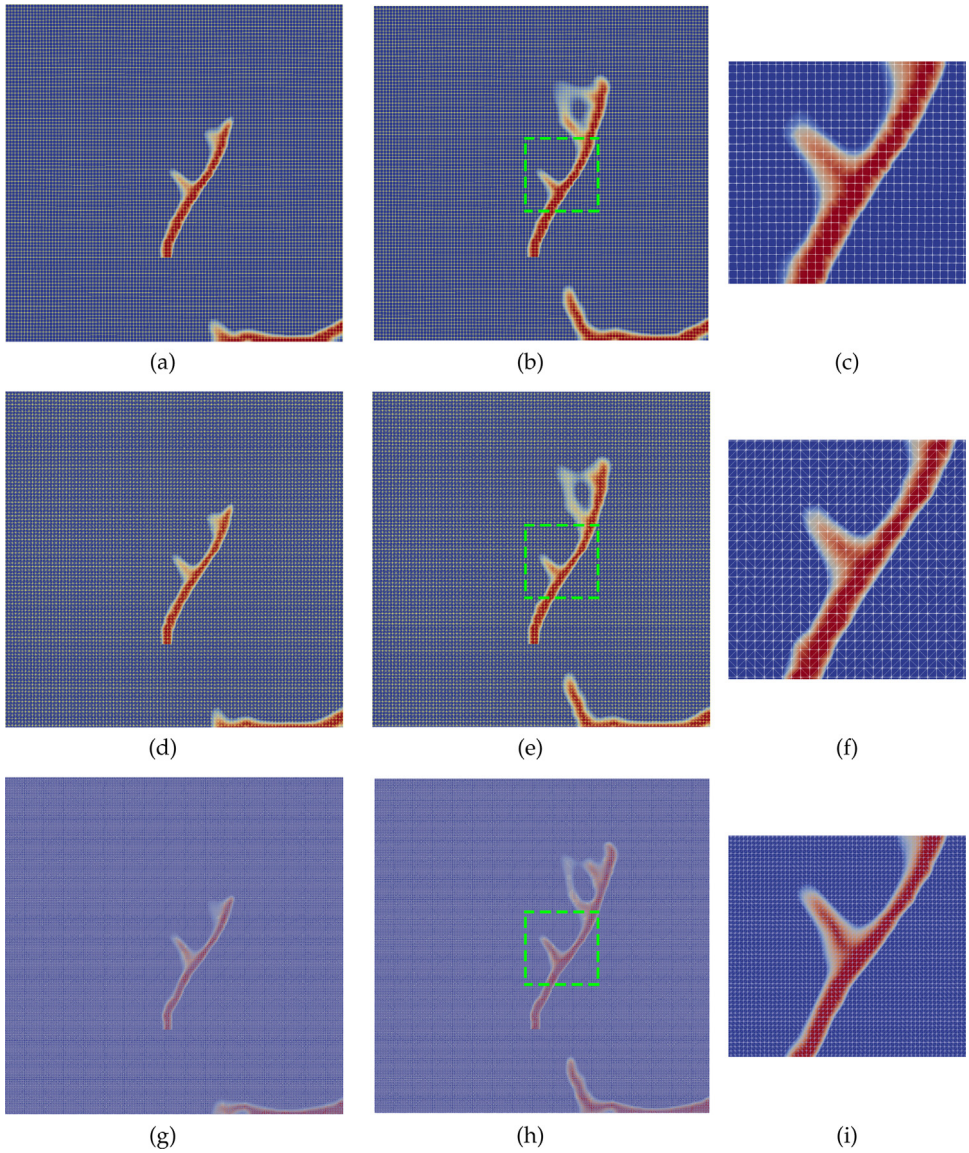


Fig. 23. Crack patterns at 65 μs (left column) and 85 μs (middle column) ($v = 33.0$ m/s) for the dynamic shear loading experiment on three different meshes: 11,450 quadrilateral elements (top row), 20,000 triangular elements (middle row), and 80,000 triangular elements (bottom row). The area around the branching at 85 μs is zoomed in to demonstrate the mesh resolution (right column).

variational constitutive update framework where discrete Lagrangian can be constructed incrementally to predict path-dependent behaviors. Furthermore, more sophisticated geometric learning methods that involve directed graphs (for representing hierarchical information), edge-weighted graphs (for representing attributes of grain contacts), and the evolution of the graphs (for path-dependent behaviors) will provide us a fuller picture to examine the relationships between topology of microstructures and the resultant macroscopic responses.

Declaration of competing interest

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

Acknowledgments

The authors would like to thank the two anonymous reviewers for their insightful feedback and suggestions that helped improve the quality of this paper. The authors are supported by the National Science Foundation, United States of America grant from the Mechanics of Materials and Structures program under grant contracts CMMI-1846875and the Office of Advanced Cyberinfrastructure under grant contracts OAC-1940203, the Dynamic Materials and Interactions Program from the Air Force Office of Scientific Research, United States of America under grant contracts FA9550-17-1-0169 and FA9550-19-1-0318. These supports are gratefully acknowledged. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the sponsors, including the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Appendix A. Graph theory terminologies and definitions

In this section, a brief review of several terms of graph theory is provided to facilitate the illustration of the concepts in this current work. More elaborate descriptions can be found in [100–102]:

Definition 1. A **graph** is a two-tuple $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ where $\mathbb{V} = \{v_1, \ldots, v_N\}$ is a non-empty **vertex set** (also referred to as nodes) and $\mathbb{E} \subseteq \mathbb{V} \times \mathbb{V}$ is an **edge set**. To define a graph, there exists a relation that associates each edge with two vertices (not necessarily distinct). These two vertices are called the edge's **endpoints**. The pair of endpoints can either be unordered or ordered (see Fig. 26).

Definition 2. An **undirected graph** is a graph whose edge set $\mathbb{E} \subseteq \mathbb{V} \times \mathbb{V}$ connects *unordered* pairs of vertices together.

Definition 3. A **loop** is an edge whose endpoint vertices are the same. When all the nodes in the graph are in a loop with themselves, the graph is referred to as allowing self-loops.

Definition 4. **Multiple edges** are edges having the same pair of endpoint vertices.

Definition 5. A **simple graph** is a graph that does not have loops or multiple edges.

Definition 6. Two vertices that are connected by an edge are referred to as **adjacent** or as **neighbors**.

Definition 7. The term **weighted graph** traditionally refers to graph that consists of edges that associate with edge-weight function $w_{ij} : \mathbb{E} \to \mathbb{R}^n$ with $(i, j) \in \mathbb{E}$ that maps all edges in $\mathbb{E}$ onto a set of real numbers. n is the total number of edge weights and each set of edge weights can be represented by a matrix $\boldsymbol{W}$ with components w_{ij}.

In this current work, unless otherwise stated, we will be referring to weighted graphs as graphs weighted at the vertices — each node carries information as a set of weights that quantify features of microstructures. All vertices are associated with a vertex-weight function $f_v : \mathbb{V} \to \mathbb{R}^D$ with $v \in \mathbb{V}$ that maps all vertices in $\mathbb{V}$ onto a set of real numbers, where D is the number of weights — features. The node weights can be represented by a $N \times D$ matrix $\boldsymbol{X}$ with components x_{ik}, where the index $i \in [1, \ldots, N]$ represents the node and the index $k \in [1, \ldots, D]$ represents the type of node weight — feature.

Definition 8. A graph whose edges are unweighted ($w_\epsilon = 1 \ \forall \epsilon \in \mathbb{E}$) can be called a **binary graph**.

To facilitate the description of graph structures, several terms for representing graphs are introduced:

Definition 9. The **adjacency matrix** $\boldsymbol{A}$ of a graph $\mathbb{G}$ is the $N \times N$ matrix whose entry α_{ij} is the number of edges in $\mathbb{G}$ with endpoints $\{v_i, v_j\}$, as shown in Eq. (3).

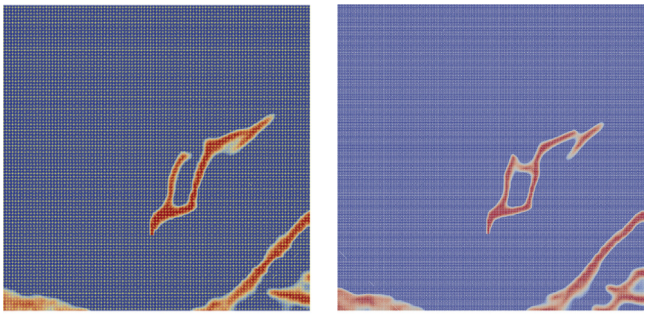


Fig. 24. Crack patterns at 70 μs ($v = 33.0$ m/s) for the dynamic shear loading experiment for the RVE rotated at $\phi = 60°$ on two different meshes: 20,000 triangular elements (left), and 80,000 triangular elements (right).

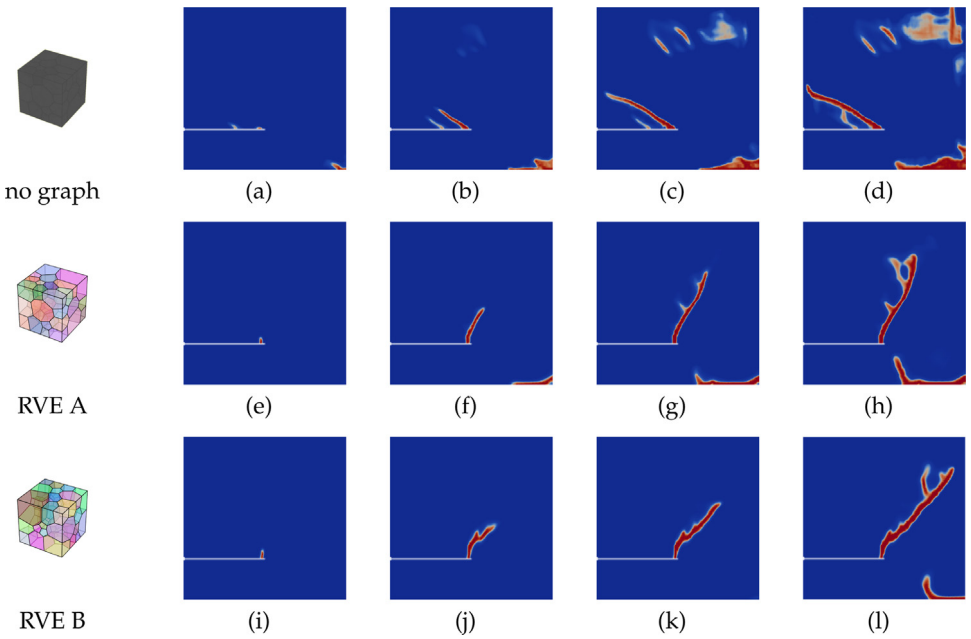


Fig. 25. Crack patterns at 30 μs, 50 μs, 65 μs, 85 μs for the dynamic shear loading experiment with an impact velocity of $v = 33.0$ m/s for a model without a graph input (a, b, c, d) and two different polycrystals (e, f, g, h and i, j, k, l). It is noted that all the parameters are identical for all the simulations but the graph input.

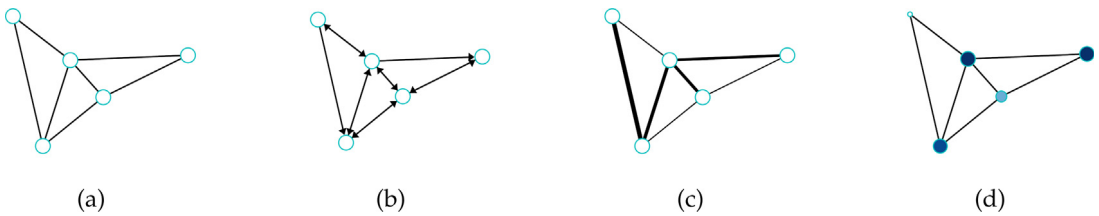


Fig. 26. Different types of graphs. (a) Undirected (simple) binary graph (b) Directed binary graph (c) Edge-weighted undirected graph (d) Node-weighted undirected graph.

Definition 10. If the vertex v is an endpoint of edge ϵ, then v and ϵ are **incident**. The **degree** d of a vertex v is the number of incident edges. The **degree matrix** $\boldsymbol{D}$ of a graph $\mathbb{G}$ is the $N \times N$ diagonal matrix with diagonal entries d_i equal to the degree of vertex v_i, as shown in Eq. (4).

Definition 11. An **isomorphism** from a graph $\mathbb{G}$ to another graph $\mathbb{H}$ is a bijection m that maps $\mathbb{V}(\mathbb{G})$ to $\mathbb{V}(\mathbb{H})$ and $\mathbb{E}(\mathbb{G})$ to $\mathbb{E}(\mathbb{H})$ such that each edge of $\mathbb{G}$ with endpoints u and v is mapped to an edge with endpoints $m(u)$ and $m(v)$. Applying the same permutation to both the rows and the columns of the adjacency matrix of graph $\mathbb{G}$ results to the adjacency matrix of an isomorphic graph $\mathbb{H}$.

Definition 12. The **unnormalized Laplacian operator $\boldsymbol{\Delta}$** is defined such that:

$$(\boldsymbol{\Delta} f)_i = \sum_{j:(i,j)\in\mathbb{E}} w_{ij}(f_i - f_j) \tag{61}$$

$$= f_i \sum_{j:(i,j)\in\mathbb{E}} w_{ij} - \sum_{j:(i,j)\in\mathbb{E}} w_{ij} f_j. \tag{62}$$

By writing the equation above in matrix form, the unnormalized Laplacian matrix $\boldsymbol{\Delta}$ of a graph $\mathbb{G}$ is the $N \times N$ positive semi-definite matrix defined as $\boldsymbol{\Delta} = \boldsymbol{D} - \boldsymbol{W}$.

In this current work, binary graphs will be used, thus, the equivalent expression is used for the unnormalized Laplacian matrix $\boldsymbol{L}$, defined as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ with the entries l_{ij} calculated as:

$$l_{ij} = \begin{cases} d_i, & i = j \\ -1, & i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0, & \text{otherwise.} \end{cases} \tag{63}$$

Definition 13. For binary graphs, the **symmetric normalized Laplacian matrix $\boldsymbol{L}^{\text{sym}}$** of a graph $\mathbb{G}$ is the $N \times N$ matrix defined as:

$$\boldsymbol{L}^{\text{sym}} = \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{L} \boldsymbol{D}^{-\frac{1}{2}} = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{A} \boldsymbol{D}^{-\frac{1}{2}}. \tag{64}$$

The entries l_{ij}^{sym} of the matrix $\boldsymbol{L}^{\text{sym}}$ are shown in Eq. (5).

Appendix B. Sample problem of graph representation of polycrystal microstructures

To demonstrate how graphs used to represent a polycrystalline assemble are generated, we introduce a simple example where an assembly consists of 5 crystals shown in Fig. 2(a) is converted into a node-weighted graph. Each node of the graph represents a crystal. An edge is defined between two nodes if they are connected/share a surface. The graph is undirected meaning that there is no direction specified for the edges. The vertex set $\mathbb{V}$ and edge set $\mathbb{E}$ for this specific graph are $\mathbb{V} = \{v_1, v_2, v_3, v_4, v_5\}$ and $\mathbb{E} = \{e_{12}, e_{23}, e_{34}, e_{35}, e_{45}\}$ respectively.

An undirected graph can be represented by an adjacency matrix $\boldsymbol{A}$ (cf. Definition 9) that holds information for the connectivity of the nodes. The entries of the adjacency matrix $\boldsymbol{A}$, in this case, are binary — each entry of the matrix is 0 if an edge does not exist between two nodes and 1 if it does. Thus, for the example in Fig. 2, crystals 1 and 2 are connected so the entries (1, 2) and (2, 1) of the matrix $\boldsymbol{A}$ would be 1, while crystals 1 and 3 are not so the entries (1, 3) and (3, 1) will be 0 and so on. If the graph allows self-loops, then the entries in the diagonal of the matrix are equal to 1 and the adjacency matrix with self-loops is defined as $\hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$. The complete symmetric matrices $\boldsymbol{A}$ and $\hat{\boldsymbol{A}}$ for this example will be:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ & 0 & 1 & 0 & 0 \\ & & 0 & 1 & 1 \\ & & & 0 & 1 \\ \text{sym.} & & & & 0 \end{bmatrix}, \qquad \hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ & 1 & 1 & 0 & 0 \\ & & 1 & 1 & 1 \\ & & & 1 & 1 \\ \text{sym.} & & & & 1 \end{bmatrix}. \tag{65}$$

A diagonal degree matrix $\boldsymbol{D}$ can also be useful to describe a graph representation. The degree matrix $\boldsymbol{D}$ only has diagonal terms that equal the number of neighbors of the node represented in that row. The diagonal terms can simply be calculated by summing all the entries in each row of the adjacency matrix. It is noted that, when self-loops are allowed, a node is a neighbor of itself, thus it must be added to the number of total neighbors for

each node. The degree matrix $\boldsymbol{D}$ for the example graph in Fig. 2 would be:

$$\boldsymbol{D} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}. \tag{66}$$

The polycrystal connectivity graph can be represented by its graph Laplacian matrix $\boldsymbol{L}$ — defined as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$, as well as the normalized symmetric graph Laplacian matrix $\boldsymbol{L}^{\text{sym}} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{L}\boldsymbol{D}^{-\frac{1}{2}}$. The two matrices for the example of Fig. 2 are calculated below:

$$\boldsymbol{L} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ & 2 & -1 & 0 & 0 \\ & & 3 & -1 & -1 \\ & & & 2 & -1 \\ \text{sym.} & & & & 2 \end{bmatrix}, \quad \boldsymbol{L}^{\text{sym}} = \begin{bmatrix} 1 & -\frac{\sqrt{2}}{2} & 0 & 0 & 0 \\ & 1 & -\frac{\sqrt{6}}{6} & 0 & 0 \\ & & 1 & -\frac{\sqrt{6}}{6} & -\frac{\sqrt{6}}{6} \\ & & & 1 & -\frac{1}{2} \\ \text{sym.} & & & & 1 \end{bmatrix}. \tag{67}$$

Assume that, for the example in Fig. 2, there is information available for two features A and B for each crystal in the graph that will be used as node weights — this could be the volume of each crystal, the orientations, and so on. The node weights for each crystal represented by a vertex v_i can be described as a vector, $\boldsymbol{f}_i = (f_A, f_B)$, such that each component of the vector corresponds to a feature of the ith node. The node features can all be represented in a feature matrix $\boldsymbol{X}$ where each row corresponds to a node and each column corresponds to a feature. For the example in question, the feature matrix would be:

$$\boldsymbol{X} = \begin{bmatrix} f_{A1} & f_{B1} \\ f_{A2} & f_{B2} \\ f_{A3} & f_{B3} \\ f_{A4} & f_{B4} \\ f_{A5} & f_{B5} \end{bmatrix}. \tag{68}$$

Appendix C. Database statistics

This section describes the statistics for the generated microstructures that were used for training the geometric learning based neural network model. The database consists of 150 polycrystal RVEs generated as described in Section 6.3. For every polycrystal in the database, the grain connectivity information is available in the form of adjacency matrices. For every crystal in a polycrystal RVE, there is available information on the crystal features that will be used as weights for the undirected graph input. The available node features in the data set contain information on the volume, the three Euler angles (in Bunge notation), the equivalent diameter (the diameter of the sphere with the equivalent volume as the crystal), the number of faces, the total area of the faces, the number of neighbors, and the centroid position vector of each crystal. More elaborate descriptions of the crystal features can be found in the documentation of the open-source software NEPER [47]. The distribution of the polycrystal features, separated into 100 training and 50 testing cases, is demonstrated in Fig. 27.

Appendix D. Encoded feature vector dimension

The hyperparameter space of a complex architecture is rather large to conduct a comprehensive hyperparameter search and provide a confident explanation of how the dimensions of the hybrid architecture affect the model's performance. The number of neurons of the encoded feature vector layer was one of these tuned hyperparameters. To provide an insight into how the encoded feature vector layer dimension choice was made, we are providing the results from three training experiments that we conducted while testing various architectures for the neural network through trial and error. We noticed that, in iterations of the network with lower dimensions than 9 neurons, the predictions were less accurate. For feature dimensions much higher than 9 neurons, the performance seemed to not drastically improve and, thus, they were not chosen to reduce the training time of the network. In Fig. 28, we are showing the performance results of training the hybrid neural network architecture with an encoded feature vector dimension of 1, 9, and 32 neurons.

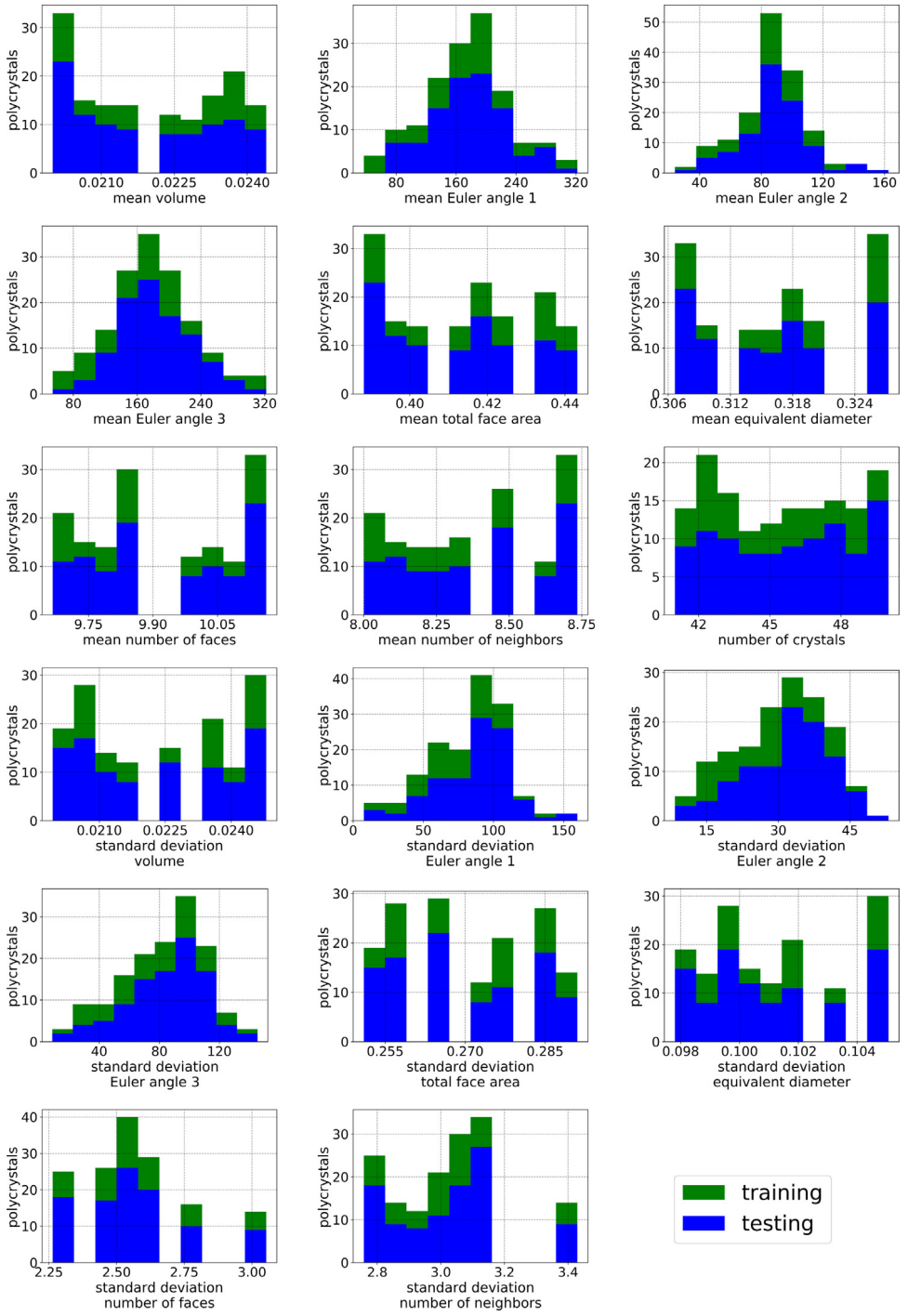


Fig. 27. Feature distributions for the 150 polycrystals in the database, separated into 100 polycrystals used for training and 50 polycrystals used for testing.

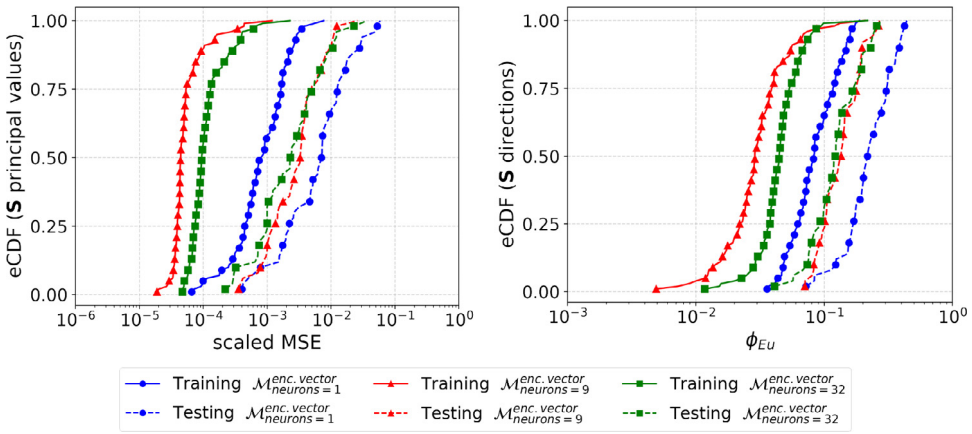


Fig. 28. Comparison of hybrid neural network performance for architectures with encoded feature vector dimensions of 1, 9, and 32 neurons.

References

[1] Arthur L. Gurson, Continuum theory of ductile rupture by void nucleation and growth: Part I—Yield criteria and flow rules for porous ductile media, J. Eng. Mater. Technol. 99 (1) (1977) 2–15.

[2] Alan Needleman, A continuum model for void nucleation by inclusion debonding, J. Appl. Mech. 54 (3) (1987) 525–531.

[3] Z.L. Zhang, C. Thaulow, J. Ødegard, A complete Gurson model approach for ductile fracture, Eng. Fract. Mech. 67 (2) (2000) 155–168.

[4] Ken Nahshon, J.W. Hutchinson, Modification of the Gurson model for shear failure, Eur. J. Mech. A Solids 27 (1) (2008) 1–17.

[5] Kim Lau Nielsen, Viggo Tvergaard, Ductile shear failure or plug failure of spot welds modelled by modified Gurson model, Eng. Fract. Mech. 77 (7) (2010) 1031–1047.

[6] Ran Ma, WaiChing Sun, Computational thermomechanics for crystalline rock. part ii: chemo-damage-plasticity and healing in strongly anisotropic polycrystals, Computer Methods in Applied Mechanics and Engineering (ISSN: 0045-7825) 369 (2020) 113184, http://dx.doi.org/10.1016/j.cma.2020.113184.

[7] Andrew Schofield, Peter Wroth, Critical State Soil Mechanics, Vol. 310, McGraw-Hill London, 1968.

[8] Ronaldo I. Borja, Seung R. Lee, Cam-clay plasticity, part 1: implicit integration of elasto-plastic constitutive relations, Comput. Methods Appl. Mech. Engrg. 78 (1) (1990) 49–72.

[9] Majid T. Manzari, Yannis F. Dafalias, A critical state two-surface plasticity model for sands, Geotechnique 47 (2) (1997) 255–272.

[10] WaiChing Sun, A unified method to predict diffuse and localized instabilities in sands, Geomech. Geoeng. 8 (2) (2013) 65–75.

[11] Yang Liu, WaiChing Sun, Jacob Fish, Determining material parameters for critical state plasticity models based on multilevel extended digital database, J. Appl. Mech. 83 (1) (2016) 011003.

[12] Kun Wang, WaiChing Sun, Simon Salager, SeonHong Na, Ghonwa Khaddour, Identifying material parameters for a micro-polar plasticity model via X-ray micro-computed tomographic (CT) images: lessons learned from the curve-fitting exercises, Int. J. Multiscale Comput. Eng. 14 (4) (2016).

[13] L. Anand, M. Kothari, A computational procedure for rate-independent crystal plasticity, J. Mech. Phys. Solids 44 (4) (1996) 525–558.

[14] SeonHong Na, WaiChing Sun, Computational thermomechanics of crystalline rock, Part I: A combined multi-phase-field/crystal plasticity approach for single crystal simulations, Comput. Methods Appl. Mech. Engrg. 338 (2018) 657–691.

[15] Ran Ma, Timothy J. Truster, Stephen B. Puplampu, Dayakar Penumadu, Investigating mechanical degradation due to fire exposure of aluminum alloy 5083 using crystal plasticity finite element method, Int. J. Solids Struct. 134 (2018) 151–160.

[16] Jean Jerphagnon, Daniel Chemla, R. Bonneville, The description of the physical properties of condensed matter using irreducible tensors, Adv. Phys. 27 (4) (1978) 609–650.

[17] WaiChing Sun, Alejandro Mota, A multiscale overlapped coupling formulation for large-deformation strain localization, Computational Mechanics 54 (2014) 803–820, http://dx.doi.org/10.1007/s00466-014-1034-0.

[18] Matthew R. Kuhn, WaiChing Sun, Qi Wang, Stress-induced anisotropy in granular materials: fabric, stiffness, and permeability, Acta Geotech. 10 (4) (2015) 399–419.

[19] Chuanqi Liu, WaiChing Sun, Ils-mpm: an implicit level-set-based material point method for frictional particulate contact mechanics of deformable particles, Comput. Methods Appl. Mech. Engrg. 369 (2020) 113168.

[20] Trenton Kirchdoerfer, Michael Ortiz, Data-driven computational mechanics, Comput. Methods Appl. Mech. Engrg. 304 (2016) 81–101.

[21] Robert Eggersmann, Trenton Kirchdoerfer, Stefanie Reese, Laurent Stainier, Michael Ortiz, Model-free data-driven inelasticity, Comput. Methods Appl. Mech. Engrg. 350 (2019) 81–99.

[22] Qizhi He, Jiun-Shyan Chen, A physics-constrained data-driven approach based on locally convex reconstruction for noisy database, 2019, arXiv preprint arXiv:1907.12651.

[23] M. Stoffel, F. Bamer, B. Markert, Stability of feed forward artificial neural networks versus nonlinear structural models in high speed deformations: A critical comparison, Arch. Mech. 71 (2) (2019).

[24] M.A. Bessa, R. Bostanabad, Z. Liu, A. Hu, Daniel W. Apley, C. Brinson, Wei Chen, Wing Kam Liu, A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality, Comput. Methods Appl. Mech. Engrg. 320 (2017) 633–667.
[25] Zeliang Liu, OrionL Kafka, Cheng Yu, Wing Kam Liu, Data-driven self-consistent clustering analysis of heterogeneous materials with crystal plasticity, in: Advances in Computational Plasticity, Springer, 2018, pp. 221–242.
[26] Nicholas Lubbers, Turab Lookman, Kipton Barros, Inferring low-dimensional microstructure representations using convolutional neural networks, Phys. Rev. E (ISSN: 2470-0045) 96 (5) (2017) 052111, http://dx.doi.org/10.1103/PhysRevE.96.052111, 2470-0053.
[27] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, ImageNet classification with deep convolutional neural networks, in: F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, Vol. 25, Curran Associates, Inc., 2012, pp. 1097–1105.
[28] Li Xu, Jimmy S.J. Ren, Ce Liu, Jiaya Jia, Deep convolutional neural network for image deconvolution, in: Advances in Neural Information Processing Systems, 2014, pp. 1790–1798.
[29] Reese E. Jones, Jeremy A. Templeton, Clay M. Sanders, Jakob T. Ostien, Machine learning models of plastic flow based on representation theory, 2018, arXiv preprint arXiv:1809.00267.
[30] A.L. Frankel, R.E. Jones, C. Alleman, J.A. Templeton, Predicting the mechanical response of oligocrystals with deep learning, Comput. Mater. Sci. (ISSN: 0927-0256) 169 (2019) 109099, http://dx.doi.org/10.1016/j.commatsci.2019.109099.
[31] Masao Satake, A discrete-mechanical approach to granular materials, Internat. J. Engrg. Sci. 30 (10) (1992) 1525–1533.
[32] WaiChing Sun, Matthew R. Kuhn, John W. Rudnicki, A multiscale dem-lbm analysis on permeability evolutions inside a dilatant shear band, Acta Geotech. 8 (5) (2013) 465–480.
[33] Antoinette Tordesillas, Sebastian Pucilowski, DavidM Walker, John F. Peters, Laura E. Walizer, Micromechanics of vortices in granular media: connection to shear bands and implications for continuum modelling of failure in geomaterials, Int. J. Numer. Anal. Methods Geomech. 38 (12) (2014) 1247–1275.
[34] Kun Wang, WaiChing Sun, Meta-modeling game for deriving theory-consistent, microstructure-based traction–separation laws via deep reinforcement learning, Comput. Methods Appl. Mech. Engrg. 346 (2019) 216–241a.
[35] Kun Wang, WaiChing Sun, An updated lagrangian lbm-dem-fem coupling model for dual-permeability fissured porous media with embedded discontinuities, Comput. Methods Appl. Mech. Engrg. 344 (2019) 276–305b.
[36] Han Altae-Tran, Bharath Ramsundar, Aneesh S. Pappu, Vijay Pande, Low data drug discovery with one-shot learning, ACS Cent. Sci. 3 (4) (2017) 283–293.
[37] Tian Xie, Jeffrey C. Grossman, Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties, Phys. Rev. Lett. 120 (14) (2018) 145301.
[38] Yoshua Bengio, Aaron Courville, Pascal Vincent, Representation learning: A review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828.
[39] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, Gabriele Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (1) (2008) 61–80.
[40] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, Jeff Dean, Distributed representations of words and phrases and their compositionality, in: C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, Vol. 26, Curran Associates, Inc., 2013, pp. 3111–3119.
[41] Aditya Grover, Jure Leskovec, Node2vec: Scalable feature learning for networks, 2016, arXiv:1607.00653 [cs, stat].
[42] Bryan Perozzi, Rami Al-Rfou, Steven Skiena, DeepWalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '14, 2014, pp. 701–710, http://dx.doi.org/10.1145/2623330.2623732.
[43] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, Shantanu Jaiswal, Graph2vec: Learning Distributed Representations of Graphs, 2017, arXiv:1707.05005 [cs].
[44] Marc'Aurelio Ranzato, FuJie Huang, Y.-Lan Boureau, Yann LeCun, Unsupervised learning of invariant feature hierarchies with applications to object recognition, in: 2007 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2007, pp. 1–8.
[45] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, Pierre-Antoine Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proceedings of the 25th International Conference on Machine Learning, ICML '08, ACM, New York, NY, USA, ISBN: 978-1-60558-205-4, 2008, pp. 1096–1103, http://dx.doi.org/10.1145/1390156.1390294.
[46] Clara Jaquet, Edward Andó, Gioacchino Viggiani, Hugues Talbot, Estimation of separating planes between touching 3d objects using power watershed, in: International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing, Springer, 2013, pp. 452–463.
[47] R. Quey, P.R. Dawson, F. Barbe, Large-scale 3d random polycrystals for the finite element method: Generation, meshing and remeshing, Comput. Methods Appl. Mech. Engrg. (ISSN: 0045-7825) 200 (17) (2011) 1729–1745, http://dx.doi.org/10.1016/j.cma.2011.01.002.
[48] Michael A. Groeber, Michael A. Jackson, Dream. 3d: a digital representation environment for the analysis of microstructure in 3d, Integr. Mater. Manuf. Innov. 3 (1) (2014) 5.
[49] Harris Drucker, ChristopherJ.C. Burges, Linda Kaufman, Alex J. Smola, Vladimir Vapnik, Support vector regression machines, in: Advances in Neural Information Processing Systems, 1997, pp. 155–161.
[50] Joaquin Quiñonero-Candela, Carl Edward Rasmussen, A unifying view of sparse approximate gaussian process regression, J. Mach. Learn. Res. 6 (Dec) (2005) 1939–1959.
[51] Jacob R. Gardner, Matt J. Kusner, Zhixiang Eddie Xu, Kilian Q. Weinberger, John P. Cunningham, Bayesian optimization with inequality constraints, in: ICML, 2014, pp. 937–945.
[52] Kun Wang, WaiChing Sun, A multiscale multi-permeability poroplasticity model linked by recursive homogenizations and deep learning, Comput. Methods Appl. Mech. Engrg. 334 (2018) 337–380.

[53] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, Philip S. Yu, Comprehensive survey on graph neural networks, 2019, arXiv: [cs, stat], arXiv:1901.00596.
[54] Michaël Defferrard, Xavier Bresson, Pierre Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: D.D. Lee, M. Sugiyama, U.V. Luxburg, I. Guyon, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 29, Curran Associates, Inc., 2016, pp. 3844–3852.
[55] Thomas N. Kipf, Max Welling, Semi-Supervised Classification with Graph Convolutional Networks, 2017, arXiv:1609.02907 [cs, stat].
[56] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE (ISSN: 0018-9219) 86 (11) (1998) 2278–2324, http://dx.doi.org/10.1109/5.726791, 1558-2256.
[57] Martin Simonovsky, Nikos Komodakis, Dynamic edge-conditioned filters in convolutional neural networks on graphs, 2017.
[58] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, Patrick Riley, Molecular graph convolutions: moving beyond fingerprints, J. Comput. Aided Mol. Des. 30 (8) (2016) 595–608.
[59] François Chollet, et al., Keras, 2015, https://keras.io.
[60] Daniele Grattarola, Spektral, 2019, URL https://danielegrattarola.github.io/spektral/.
[61] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (2014) 1929–1958.
[62] J. Ghaboussi, J.H. Garrett Jr., Xiping Wu, Knowledge-based modeling of material behavior with neural networks, J. Eng. Mech. 117 (1) (1991) 132–153.
[63] M. Lefik, D.P. Boso, B.A. Schrefler, Artificial neural networks in numerical modelling of composites, Comput. Methods Appl. Mech. Engrg. 198 (21–26) (2009) 1785–1804.
[64] Zeliang Liu, C.T. Wu, M. Koishi, A deep material network for multiscale topology learning and accelerated nonlinear modeling of heterogeneous materials, Comput. Methods Appl. Mech. Engrg. 345 (2019) 1138–1168.
[65] Xiaoxin Lu, DimitrisG Giovanis, Julien Yvonnet, Vissarion Papadopoulos, Fabrice Detrez, Jinbo Bai, A data-driven computational homogenization method based on neural networks for the nonlinear anisotropic electrical response of graphene/polymer nanocomposites, Comput. Mech. 64 (2) (2019) 307–321.
[66] Daniel Z. Huang, Kailai Xu, Charbel Farhat, Eric Darve, Predictive Modeling with Learned Constitutive Laws from Indirect Observations, 2019, arXiv preprint arXiv:1905.12530.
[67] M. Zytynski, M.F. Randolph, R. Nova, C.P. Wroth, On modelling the unloading-reloading behaviour of soils, Int. J. Numer. Anal. Methods Geomech. 2 (1) (1978) 87–93.
[68] Ronaldo I. Borja, Claudio Tamagnini, Angelo Amorosi, Coupling plasticity and energy-conserving elasticity models for clays, J. Geotech. Geoenviron. Eng. 123 (10) (1997) 948–957.
[69] Gerhard A. Holzapfel, Thomas C. Gasser, Ray W. Ogden, A new constitutive framework for arterial wall mechanics and a comparative study of material models, J. Elast. Phys. Sci. Solids (ISSN: 1573-2681) 61 (1) (2000) 1–48, http://dx.doi.org/10.1023/A:1010835316564.
[70] B.A. Le, Julien Yvonnet, Q.-C. He, Computational homogenization of nonlinear elastic materials using neural networks, Internat. J. Numer. Methods Engrg. 104 (12) (2015) 1061–1084.
[71] G.H. Teichert, A.R. Natarajan, A. Van der Ven, K. Garikipati, Machine learning materials physics: Integrable deep neural networks enable scale bridging by learning free energy functions, Comput. Methods Appl. Mech. Engrg. 353 (2019) 201–216.
[72] Gregory H. Teichert, Krishna Garikipati, Machine learning materials physics: Surrogate optimization and multi-fidelity algorithms predict precipitate morphology in an alternative to phase field dynamics, Comput. Methods Appl. Mech. Engrg. 344 (2019) 666–693.
[73] Wojciech M. Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, Razvan Pascanu, Sobolev training for neural networks, in: Advances in Neural Information Processing Systems, 2017, pp. 4278–4287.
[74] Sho Sonoda, Noboru Murata, Neural network with unbounded activation functions is universal approximator, Appl. Comput. Harmon. Anal. 43 (2) (2017) 233–268.
[75] Yousef Heider, Kun Wang, WaiChing Sun, So(3)-invariance of informed-graph-based deep neural network for anisotropic elastoplastic materials, Comput. Methods Appl. Mech. Engrg. 363 (2020) 112875.
[76] Ronaldo I Borja, Plasticity, Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN: 978-3-642-38546-9, 2013, http://dx.doi.org/10.1007/978-3-642-38547-6.
[77] Kenichi Tamura, Marcus Gallagher, Quantitative measure of nonconvexity for black-box continuous functions, Inform. Sci. 476 (2019) 64–82.
[78] Ran Ma, WaiChing Sun, Fft-based solver for higher-order and multi-phase-field fracture models applied to strongly anisotropic brittle materials and poly-crystals, Comput. Methods Appl. Mech. Engrg. (2019) tentatively accepted..
[79] Yuan-cheng Fung, Foundations of Solid Mechanics, 1965.
[80] Gerard A. Ateshian, Kevin D. Costa, A frame-invariant formulation of fung elasticity, J. Biomech. 42 (6) (2009) 781–785.
[81] F. Bachmann, Ralf Hielscher, Helmut Schaeben, Texture Analysis with MTEX – Free and Open Source Software Toolbox, 2010, http://dx.doi.org/10.4028/www.scientific.net/SSP.160.63.
[82] Maurice George Kendall, et al., The advanced theory of statistics, in: The Advanced Theory of Statistics, second ed., 1946.
[83] J.E. Gentle, Computational Statistics, Springer, ISBN: 978-0-387-98145-1, 2009.
[84] Du Q. Huynh, Metrics for 3d rotations: Comparison and analysis, J. Math. Imaging Vision 35 (2) (2009) 155–164.
[85] K.H. Roscoe, J.B. Burland, On the generalized stress–strain behaviour of wet clay, 1968.
[86] G.T. Houlsby, The use of a variable shear modulus in elastic–plastic models for clays, Comput. Geotech. 1 (1) (1985) 3–13.
[87] Ronaldo I. Borja, Chao-Hua Lin, Francisco J. Montáns, Cam-clay plasticity, part iv: Implicit integration of anisotropic bounding surface model with nonlinear hyperelasticity and ellipsoidal loading function, Comput. Methods Appl. Mech. Engrg. 190 (26–27) (2001) 3293–3323.

[88] Yoshua Bengio, Yves Grandvalet, No unbiased estimator of the variance of k-fold cross-validation, J. Mach. Learn. Res. 5 (Sep) (2004) 1089–1105.
[89] J.F. Kalthoff, S. Winkler, Failure mode transition at high rates of shear loading. DGM informationsgesellschaft mbH, Impact Loading Dyn. Behav. Mater. 1 (1988) 185–195.
[90] Joerg F. Kalthoff, Modes of dynamic shear failure in solids, Int. J. Fract. (ISSN: 1573-2673) 101 (1) (2000) 1–31, http://dx.doi.org/10.1023/A:1007647800529.
[91] Juan C. Simo, J.W. Ju, Strain-and stress-based continuum damage models—i. formulation, Int. J. Solids Struct. 23 (7) (1987) 821–840.
[92] Julian Kochmann, Lisa Ehle, Stephan Wulfinghoff, Joachim Mayer, Bob Svendsen, Stefanie Reese, Efficient multiscale fe-fft-based modeling and simulation of macroscopic deformation processes with non-linear heterogeneous microstructures, in: Multiscale Modeling of Heterogeneous Structures, Springer, 2018, pp. 129–146.
[93] Blaise Bourdin, Gilles A. Francfort, Jean-Jacques Marigo, The variational approach to fracture, J. Elasticity 91 (1–3) (2008) 5–148.
[94] Michael J. Borden, Clemens V. Verhoosel, Michael A. Scott, Thomas J.R. Hughes, Chad M. Landis, A phase-field description of dynamic brittle fracture, Comput. Methods Appl. Mech. Engrg. 217 (2012) 77–95a.
[95] Christian Miehe, Martina Hofacker, Fabian Welschinger, A phase field model for rate-independent crack propagation: Robust algorithmic implementation based on operator splits, Comput. Methods Appl. Mech. Engrg. 199 (45) (2010) 2765–2778b.
[96] Eduardo A de Souza Neto, Djordje Peric, David R.J. Owen, Computational Methods for Plasticity: Theory and Applications, John Wiley & Sons, 2011.
[97] Ted Belytschko, Hao Chen, Jingxiao Xu, Goangseup Zi, Dynamic crack propagation based on loss of hyperbolicity and a new discontinuous enrichment, Internat. J. Numer. Methods Engrg. (ISSN: 1097-0207) 58 (12) (2003) 1873–1905, http://dx.doi.org/10.1002/nme.941.
[98] Jeong-Hoon Song, Hongwu Wang, Ted Belytschko, A comparative study on finite element methods for dynamic fracture, Comput. Mech. (ISSN: 1432-0924) 42 (2) (2008) 239–250, http://dx.doi.org/10.1007/s00466-007-0210-x.
[99] Haoyan Wei, Jiun-Shyan Chen, A damage particle method for smeared modeling of brittle fracture, Int. J. Multiscale Comput. Eng. 16 (4) (2018).
[100] Ronald L. Graham, Donald E. Knuth, Oren Patashnik, Stanley Liu, Concrete mathematics: a foundation for computer science, Comput. Phys. 3 (5) (1989) 106–107.
[101] Douglas Brent West, et al., Introduction to Graph Theory, Vol. 2, Prentice hall Upper Saddle River, 2001.
[102] Jørgen Bang-Jensen, Gregory Z. Gutin, Digraphs: Theory, Algorithms and Applications, Springer Science & Business Media, 2008.