Deep learning and multi-level featurization of graph representations of microstructural data

Reese Jones · Cosmin Safta · Ari Frankel

Abstract

Many material response functions depend strongly on microstructure, such as inhomogeneities in phase or orientation. Homogenization presents the task of predicting the mean response of a sample of the microstructure to external loading for use in subgrid models and structure-property explorations. In contrast to mathematically derived homogenization techniques, data driven machine learning can provide empirical global average quantities of interest where traditional homogenization theory is lacking. Although many microstructural fields have obvious segmentations, learning directly from the graph induced by the segmentation can be difficult because this representation does not encode all the information of the full field. We develop a means of deep learning of hidden features on the reduced graph given the native discretization and a segmentation of the initial input field. The features are associated with regions represented as nodes on the reduced graph. This reduced representation is then the basis for the subsequent multi-level/scale graph convolutional network model. There are a number of advantages of reducing the graph before fully processing it with convolutional layers, such as interpretable features and efficiency on large meshes. We demonstrate the performance of the proposed network relative to convolutional neural networks operating directly on the native discretization of the data using three physical exemplars.

Keywords Graph convolutional neural networks · Inelasticity · Microstructure response · Multilevel graphs

1 Introduction

Newly developed graph neural networks (GNNs) [1–3], in particular convolutional graph neural networks, have been shown to be effective in a variety of classification and regression tasks. Recently they have been applied to physical problems [4,5] where they can accommodate unstructured and hierarchical data naturally. Analogous to pixel-based convolutional neural networks (CNNs), “message passing” [6] graph convolutional neural networks (GCNNs) [7,8] employ convolutional operations to achieve a compact parameter space by exploiting correlations in the data through connectivity defined by adjacency on the source discretization. Frankel et al. [9] and others [10–13] derive the information transmission graph directly from the connectivity of the discretization, computational grid or mesh based on the assumption the physical interactions are local. Some obvious advantages of applying convolutions to the discretization graph are that: general mesh data can be handled without interpolation to a structured grid, the discretization can be conformal to the microstructure, periodic boundary conditions can be handled without padding, and topological irregularities can be accommodated without approximations. In this approach the kernels and number of parameters are similar for a pre-selected reduction of the representation, e.g. based on the grains in a polycrystal [5], but the size of the adjacency can be prohibitive. Naively constructed and applied, these mesh-based graph models can operate on very large graphs; however, graph reduction is important for efficiency and can promote learning [14]. While physical problems with short- (e.g. interface) and long-range (e.g. elastic) interactions are ubiquitous in engineering and materials science, treating the longer scales with convolutions on the discretization graph can be inefficient or ineffective. In this work we focus on providing GNN representations of the homogenization problem where microstructural information with short and long range influence is aggregated to predict sample-level quantities of interest. This empirical, implicit form of homogenization does not have the mathematical underpinnings of classical homogenization theory [15–17],
nor its direct interpretability, but provides utility where the theory is lacking.

Treating aggregated data via graph pooling based on data clustering is a long-standing field of research with broad applications, e.g., image classification [18], mesh decomposition [19], and chemical structure [20]. Since clustered data rarely has a regular topology, graph-based networks are natural representations. In particular, compared to a CNN operating on a pixelized image, a GCNN operating on an unstructured mesh has less well-defined spatial locality and inherits the varying neighborhood size of the source data. Akin to topologically localized convolutions supplanting spectral-based convolutions [7,21] and spectral pre-processing [22] in the literature, spectral clustering based on the primary eigenvectors of the graph Laplacian has been largely superseded by less expensive, more easily differentiable techniques, some of which connect to spectral clustering. Dhillon et al. [23] showed the equivalence of spectral clustering and kernel k-means clustering, which allowed expensive eigenvalue problems to be reframed in terms of trace maximization objectives. These objectives include maximizing in-cluster links/edges and minimizing cuts, i.e., number of links, between any cluster and remainder of the graph. Typically these objectives are normalized relative to cluster size (number of nodes) or degree (sum of non-self connections/adjacency) but are amenable to the data on the graph. More recently, graph-based neural nets, such as DiffPool [24] and MinCutPool [14], have been designed to take into account the data on the graph in soft clustering, trainable pooling operations. Ying et al. [24] developed DiffPool to enable a hierarchical treatment of graph structure. DiffPool uses a GNN for the ultimate classification task and another with a softmax output for the intermediary pooling task. The separate GNNs learn a soft, in the sense not binary and disjoint, assignment of nodes in the input graph to those in a smaller graph as well as derivative features on the smaller embedded graph. Due to non-convexity of the clustering objective, an auxiliary entropy loss is used to regularize the training. Bianchi, Grattarola and Alippi [14] developed MinCutPool based on a degree normalized objective of minimizing edges between any cluster and the remainder of the graph. They relaxed the objective of finding a binary, disjoint cluster assignment to reduce the computational complexity by recasting the problem to a continuous analog. Ultimately they proposed a network similar to DiffPool albeit with the softmax assignment GNN replaced by a softmax multilayer perceptron (MLP) and different loss augmentation designed to promote orthogonality of the clustering matrix mapping graph nodes to clusters. Grattarola et al. [25] also generalized the myriad approaches to pooling and clustering on graphs with their select-reduced-connect abstraction of these operations.

Graph convolutional neural networks can be particularly opaque in how they achieve accurate models. To interpret what convolution networks are learning through their representations, filter visualization and activation [26–28], saliency maps [29], sensitivity, attention based and related techniques [30,31] have been developed. These techniques have been largely applied to the ubiquitous commercial classification tasks, for instance Yosinski et al. [28] demonstrated how activations at the deepest layers of a CNN correspond to obvious features in the original image used in classification, e.g., faces. Some of these techniques have enabled a degree of filter interpretation by translating the filter output so that the learned features are obvious by inspection. For pixel-based CNNs deconvolution and inversion of the input–output map [27], as well as guided/regularized optimization [28] have produces some insights. Some other methods rely on clustering, for example, Local Interpretable Model-agnostic Explanations (LIME) [32] is based on segmentation and perturbation of the cluster values. In general, clustering simplifies the input–output map created by the convolution filter and therefore the complexity of the image-to-image convolutional transformation.

Given that in many physical homogenization problems the segmentation of the domain based on specific features, e.g., phase or orientation, of the constituents is readily accessible but the informative features on the sub-domains are not, we propose a means of constructing a graph-based representation of the average quantities of interest where the features on the nodes representing the sub-domains are learned/not pre-selected while the clustered topology is known. Aspects of the proposed methodology resembles the reduction-prolongation operations of algebraic multi-grid approaches [33], the architecture of graph U-nets [34] and the combination of multigrid with pixel-based CNNs [35]. This concept is extensible to multiple levels for data that naturally resides in unstructured meshes through interscale graph operators, as we will demonstrate. Unlike recent developments where GCNNs are applied inherently discrete systems [4,36,37] our methodology is adapted to continuum applications. Beyond developing a GCNN architecture to efficiently reduce data on the native discretization graph to logical clustering without manual feature engineering, a secondary goal is to provide some insight into how the resulting models learn an accurate representation of the data.

This paper is organized as follows. In Sect. 2 we describe the homogenization problems of interest. Sect. 3 describes the proposed architecture and contrast it with traditional CNN and GCNN-based architectures. Sect. 4 outlines the techniques we use to interpret the flow of information in the networks. In Sect. 5 we present the physical exemplars we model with GCNNs. Then in Sect. 6 we demonstrate the performance and interpretation of the proposed framework on these exemplars. With the idea that the simplest accu-
rate model is likely to be the most interpretable, we compare the proposed deep reduced GCNN to analogous architectures based on previous GCNN [9] and CNN [38] representations. We conclude in Sect. 7 with a discussion of the proposed architecture in the context to soft, learned clustering, and extensions left for future work.

2 Homogenization

Homogenization, in the sense of obtaining averages of the physical response of a sample with complex structure, represents a class of problems with broad physical applications, such as creating sub-grid models of materials with multiple phases, orientations or other microstructural aspects. For each particular physical problem the input is a sample of the detailed structure in a regular domain and the output is the mean (or total) response of the sample subject to external loading. Samples that are of a size that is representative of the large sample limit are called representative volume elements (RVEs) and those that are smaller, where statistical variations are apparent in the output, are called statistical volume elements (SVEs) [16,17]; this distinction is clearly a matter of degree.

For example, the volume average of stress evolution \( \sigma(t) \) as a function of (boundary loading) strain \( \tilde{\epsilon}(t) \) is often of interest in developing material models. The boundary value problem over the sample gives a field, in this example stress \( \sigma(X,t) \), as a function of position \( X \) and time \( t \):

\[
\sigma(X,t) = \sigma(\phi(X), \tilde{\epsilon}(t)),
\]

where \( \phi(X) \) is a field describing the initial microstructure, and \( \tilde{\epsilon}(t) \) is an external loading applied through the boundary conditions. Homogenization aims to represent the average response over the sample domain \( \Omega \):

\[
\bar{\sigma}(t) = \frac{1}{V} \int_{\Omega} \sigma(X,t) \, dV
\]

as a function of \( \phi(X) \) and \( \tilde{\epsilon}(t) \). Here \( V \) is the volume of the sample domain \( \Omega \). The field \( \phi(X) \) can be interpreted as a (potentially multichannel) image resulting from, for instance, computed tomography and/or electron backscatter diffraction (refer to Fig. 1). For this work we assume that \( \phi(X) \) over the sample domain \( \Omega \) can be readily segmented into disjoint regions \( \Omega_K \) that each have a uniform value \( \phi_K \), as in Fig. 1. The discretization necessary to resolve the fields leads to the number of discretization cells \( N_{\text{cells}} \) being much greater than the number of regions \( N_{\text{clusters}} \) in general.

Theory-based homogenization traditionally has relied on analytical models based on simplifying assumptions. Mixture rules where individual contributions of the constituents of the microstructure are summed and weighted by their volume fractions, \( \chi_K = V_K / V \), have generally been utilized in approximate analytical homogenization. For instance, the approximation

\[
\bar{\sigma} = \frac{1}{V} \sum_K \int_{\Omega_K} \sigma \, dV \approx \sum_K \chi_K \sigma_K
\]

(3)

where \( \sigma_K = \sigma(\phi_K, \epsilon_K) \) is widely employed. Note that mixture rules such as these typically pair extensive partitioning variables, such as \( \chi_K \), with intensive variables, such as \( \phi_K \). The approximation in Eq. (3) leads to an estimate of the elastic moduli \( C \equiv \partial_\epsilon \sigma \):}

\[
\bar{C} \approx \sum_K \chi_K C_K.
\]

(4)

assuming the strain \( \epsilon \) is homogeneous in the sample. Homogeneous flux or homogeneous field gradient are typical simplifying assumptions for homogenizing gradient-driven fluxes such as stress and heat flux. These complementary limiting cases can be combined, as in the classic Hill average estimator [39] for the elastic modulus tensor:

\[
\bar{C} = \frac{1}{2} \left( \sum_K \chi_K C_K + \left[ \sum_K \chi_K C_K^{-1} \right]^{-1} \right).
\]

(5)

Note that assuming uniform stress or strain in all \( \Omega_K \) in the domain \( \Omega \) omits compatibility requirements that make the actual field dependent on differences in \( \phi_K \) along interface (e.g. orientation differences along inter-crystal boundaries [misorientation] [40]). Without incurring the expense of solving the boundary value problem these type of limitations are difficult to surmount with theory. The goal of our data driven approach is to overcome this limitation with an empirical neural network model.

3 Graph neural network architecture

In this section we describe the proposed multi-level graph neural network for deep learning of homogenized response using segmented images of initial microstructure. The network we propose is deep to address the question of what features need to be associated with the nodes of the cluster-level graph in order to create an accurate model. Unlike pixelized images used in CNNs, graph representations of clustered image data typically cannot be reconstructed into images and therefore are incomplete embeddings of the information in the input images. This is even true of graphs that embed distances in their edge features or adjacency matrices; and, clearly, the usual binary adjacency induces a different sense of distant between nodes than in the original image.
Some features beyond the image values in the subdomains are obvious, such as the cell volumes used in mixture homogenization formulae; whereas, others are not and discovering the hidden features will be the task of the proposed deep neural network. In the proposed architecture we also seek to take advantage of multiple physical scales to interactions, e.g. short-range interactions at the level of the discretization and longer range at the level of the clusters. By combining information at multiple levels in a deep manner we expect to produce a more accurate model.

In our previous work [9], we showed that GCNNs based on the native discretization of the image data, whether on unstructured meshes or structured grids, provided an effective representation for homogenization tasks. This approach can become prohibitive for very fine discretizations since the adjacency matrices and node data scale with the size of the discretization. Unlike graph clustering solely based on node adjacency [23] independent of data on graph, here the initial data \( \phi \) clearly informs the clustering of the adjacency derived from the computational mesh. So we take an approach similar to the soft assignment GNNs discussed in Sect. 1 [24,41]; however, we provide a cluster assignment matrix of the mesh entities based on SVE subdomains as data.

Restricting our multi-level conception of the graph appropriate for microstructure homogenization problem to three levels: discretization, cluster, and sample, a general framework is suggested by Fig. 2. In this schematic inputs on the left are transmitted to outputs on right through a sequence of operations. Inputs can be known features at the discretization/mesh (e.g. element volumes), cluster (e.g. phases), or sample/global level (e.g. volume fraction). Convolution (black horizontal arrows) with trainable filters extract informative features correlated with output of interest and maintains the level of the nodal data. The reduction (downward blue arrows) and prolongation (upward red arrows) operations exchange information at levels. Figure 3 illustrates four specific variants of the framework. The vGCNN (vGCNN, Fig. 3a) is an example where information can flow cluster to discretization and vice versa. Whereas, in the down variant (dGCNN, Fig. 3c) only finer to coarser information flow is allowed (discretization to cluster and cluster to global). The reduced GCNN (rGCNN, Fig. 3d) is a simplified version where only one convolution at discretization is prescribed. (The reasons for this design discussed in more detail later in this section.) For comparison, the GCNN operating at the full discretization level [9] (fGCNN, Fig. 3b) is also shown.

Taking the rGCNN as an example, a more detailed schematic is shown in Fig. 4. The inputs for the architecture, are the discretization-level node feature matrix \( \phi \) of selected features on image/computational cells, the adjacency matrix \( A \) formed from the topology of the native discretization (mesh), and a clustering assignment matrix \( S \) determined by the SVE subdomain segmentation. Each input is unique to the particular SVE sample and consists, for example, of the obvious features from imaging: phase, orientation and from the discretization: volume of the pixel or element. The adjacency matrix \( A \) is constructed directly from the mesh connectivity and it is a symmetric binary matrix \( N_{\text{cells}} \times N_{\text{cells}} \) with ones for row \( i \) and column \( j \) where cells \( i \) and \( j \) are considered neighbors and zeros elsewhere:

\[
A_{ij} = \begin{cases} 
1 & \text{if nodes } i, j \text{ are neighbors} \\
0 & \text{else} 
\end{cases} \quad (6)
\]

The (known) data segmentation as encoded by a \( N_{\text{clusters}} \times N_{\text{cells}} \) matrix \( S \) which maps elements to clusters in one-hot type encoding such that any row of \( S \) has only one “1” and
Fig. 3  Multilevel GCNN architecture variants

For instance $A_0$ describes self interactions and $A_1$ describes first nearest neighbors. As in Ref. [9] we use a 2 weight GCNN (independent self and neighbor weights), as opposed to the commonly employed Kipf and Welling convolution that makes the self weight dependent on the neighbor weight [8].

Clustering, as guided pooling, reduces the full adjacency $A$ to the adjacency on the reduced graph $A^* = S^T A S \ (12)$ subject to subsequent normalization. The on-diagonal components scale like $A^*_I = \sum_{i,j \in \Omega_I, i \neq j} \bar{n} N_I 1 \propto V_I$ and the off-diagonal like $A^*_{IJ} = \sum_{i \in \Omega_I, j \in \Omega_J} 1 \approx 2(\bar{n}/2)N_{IJ} \propto A_{IJ}$ where $\bar{n}$ is the average number of neighbors, $V_I$ is the volume of the cluster, and $A_{IJ}$ is the surface area between clusters $I$ and $J$. The reduced graph adjacency clearly encodes relevant physical information about the SVE. The discrepancies in the different scaling of the diagonal and off-diagonal entries are handled by the two independent trainable weights.

The assignment matrix $S$ also plays a role in generating the node features $Z^*$ on the reduced graph but it can be ineffective to merely restrict the input data $Z^* = S^T Z$ (see Ref. [9] and Sect. 6.1), instead they are attained by a trainable convolution

$$Z^* = S^T \text{Conv}(Z, A) \ (13)$$

as in Ref. [24]. This layer (or layers) does the task of featurization, i.e. determining the combinations of data from the source image that are needed to create an effective model of the output. The number of filters of this layer $N_{\text{filters}}$ limits the number of features $N_{\text{features}}$ of the node data $Z^*$ on the reduced graph; however, as we will demonstrate in Sect. 6.2 this is not a significant hindrance since selecting $N_{\text{filters}}$ greater than the necessary number of features leads to a model with similar performance.

\[ SK_i = \begin{cases} 1 & \text{if node } i \text{ is in cluster } \Omega_K \\ 0 & \text{else} \end{cases} \quad (7) \]

where the columns are assigned to regions $\Omega_K$ in no particular order. The matrix $S$ acts as a restriction by mapping a vector in the full/mesh space to the reduced and $S^T$ acts as a prolongation by a vector in the reduced/cluster space to the full. For instance, the orientation on the full graph is given by

$$\phi = S^T \phi^* \quad (8)$$

where $\phi^*$ is a vector of orientation assignments per region $\Omega_K$; and, conversely, the vector of cluster volumes is given by

$$V^* = SV \quad (9)$$

where $V$ is a vector of element volumes.

As in our previous work [9,42], the core of the model is based on graph convolution which can be expressed as

$$Z' = \text{Conv}(Z, A) \equiv a(WZ + b) \quad (10)$$

where $a$ is a non-linear activation function applied element-wise, $W$ is a trainable weight matrix and $b$ is a trainable bias vector. Here we use ReLU activations based on previous work [9]. The weight matrix $W$ is composed of the sum of products of $w_K$ trainable scalars and $A_K$ pre-defined adjacency matrices:

$$W = \sum_K w_K A_K \ , \quad (11)$$
performance to a tuned one. We expect $N_{\text{filters}} \geq N_{\text{channels}}$ to encode information lost in the graph embedding.

Since one of the objectives is to achieve comparable performance to a GCNN operating solely on graph representations of the mesh discretization [9], we examine the post-featurization convolution (refer to Fig. 4):

$$\text{Conv}(Z^a, A^*) = W^a Z^a + b^* = SWS^T SZ + b^*$$

which would be the reduction of the convolution on the full graph $WZ + b$ if $S^T SZ = Z$. Here, $W$ and $b$ the weights and biases of the convolution on the full graph, and $W^a$ and $b^*$ the weights and biases on the corresponding reduced graph. The assignment matrix is such that

$$\sum_k S_{ik} S_{jk} = \sum_k S^2_{ik} \delta_{IJ} = N_I \delta_{IJ} ,$$

where $N_I$ is the number of cells in cluster $I$, so we can normalize this product by the cluster sizes. This normalization effects a cluster-wise average. In analog to global pooling operations (sum, average, min, max) other cluster level pooling can be effected with difference normalizations. We found that the reduction operations layer that process the inputs $\phi$ it was best to use sum-like reductions that result in a total for extensive inputs like volumes, and average reductions for intensive variables like orientation. Note the equivalence between the convolution on the full graph and the reduced graph only holds when the node data $Z$ is uniform in the clusters, which is true for the input $Z$ but not after subsequent convolutions. In this work we only employ one featurizing convolution Eq. (13) with nearest neighbor adjacencies; however, a stack of 1-hop convolutions or a $k$-hop convolutions could be employed.

The remainder of the network operates on the reduced graph and associated features and culminates in a global pooling to create system-level features. Our previous work [9] and others [14] has indicated that a skip connection with an independent self weight is beneficial, so we use graph convolutions with 2 trainable $w_K$ as in Ref. [9]. Since the number of output features is determined by the number of filters in this convolutional stack, we denote them as $N_{\text{features}}$. The final global pooling/average shown in Fig. 4

$$\text{Pool}(Z) = \frac{1}{N_{\text{cells}}} \sum_i Z_i$$

accounts for different number of clusters differ across samples and provides global scalar features $F_a$ that are mixed by a linear dense layer to be used as input to an MLP for property prediction, for example effective conductivity Sect. 5.1, or a recurrent neural network for evolution prediction, as with the crystal plasticity exemplary Sect. 5.2.

The architecture presented above was implemented via TensorFlow [43] and Spektral [44].

## 4 Interpretation methods

By design in the proposed architecture features $F_a$ are equivalent to the average of the final filter outputs, refer to Fig. 4, in a loose analogy with traditional homogenization estimators discussed in Sect. 2. We can correlate these features $F_a$ with the physical outputs of the network, e.g. $\sigma(t)$

$$\rho_{F_a, \sigma} = \frac{\text{Var}[F_a, \sigma]}{\sqrt{\text{Var}[F_a]} \sqrt{\text{Var}[\sigma]}}$$

to determine what regimes are most active. Here, $\text{Var}[x]$ is the variance of $x$ and $\text{Cov}[x, y]$ is the covariance of $x$ and $y$. Filter activation can be assessed by the statistics of whether the filter of a particular convolutional layer produces non-zero output over the ensemble of input images $\phi(X)$. The activation may change over different regimes in the response $\sigma(\epsilon)$, which can indicate specialization to, for example, the elastic response or plastic flow. Simple filter visualization pairing the output image given an input image can also be illuminating since it can show what regions activate the filter. (In the case of a GCNN the map from the graph output to an image is based on node assignments to source mesh.) Filter maps of the input $\phi(X)$ to the filter output of the last convolutional layer $X_a(i, a)$, corresponding to the global pooled feature $F_a$ can also be constructed and visualized with density plots. Note that the redundancy that is common in NNs confounds interpretation, so to aid interpretability with the architecture shown in Fig. 4 we simplify the previously proposed network architecture [9] by omitting dense layers.

## 5 Datasets

To demonstrate the efficacy of the proposed architecture we will use a few exemplars: (a) heat conduction in polycrystals, (b) stress evolution in polycrystalline materials undergoing plastic deformation, and (c) stress evolution in viscoelastic composite materials. The aspects of the data germane to machine learning are described in the following section and the remaining simulation details are given in Appendix A. With these exemplars we generate both 2D and 3D datasets. For each exemplar only the apparent orientation or phase information and the discretization cell volumes are provided to the GCNN models. The inclusion of cell volumes is motivated by the classical mixture formulas since the unstructured meshes have a distribution of element volumes. The 3D
datasets demonstrate the full generality of the neural network architectures described in Sect. 3 and test how they perform on samples with large discretizations, while the 2D datasets facilitate exploration through visualization. Some of the datasets involve data pixelated on structured grids, which are treated as graphs by the graph convolutional neural networks. These datasets enable direct comparison with pixel-based CNNs.

5.1 Polycrystal heat conduction

The simplest dataset was generated with two-dimensional simulations of steady heat conduction through ersatz polycrystals represented on a triangular mesh. The problem has the complexity that the conductivity tensor \( \kappa \) in each crystal is orientation \( \phi \) dependent; however, the governing partial differential equation is linear in the conductivity and the temperature fields. Figure 5 shows a few of the 10,000 realizations generated, which had 211–333 elements (mean 235.8) and 14–18 crystals (mean 15.4). Also apparent is deviation in the temperature fields from homogeneous gradients due to the anisotropy of the crystal components of the samples.

As Fig. 6a shows, the volume averaged angle \( \bar{\phi} \) for a realization is not particularly correlated with the output. However if the crystal conductivities are known, mixture rules analogous to Eq. (5) for a homogeneous gradient

\[
\bar{\kappa} = \sum_K \chi_{KK} \kappa
\]  

(18)

give reasonably well-correlated estimates of the effective conductivity \( \bar{\kappa} \). These estimates are show in Fig. 6b for each of the realizations in the ensemble. Clearly the mixture estimates are biased since the assumptions of homogeneous gradient or homogeneous flux are limiting cases.

5.2 Polycrystals with crystal plasticity

Crystal plasticity (CP) is another common example where the homogenized response of a representative sample of grain structure characterized by the orientation angles \( \phi(X) \) is of interest. As detailed in Appendix A.2, the response of the polycrystalline metal samples is complex and non-linear in the loading and properties. Figure 7 shows a few of the 12,000 2D realizations on a 32x32 grid and another 10,000 3D realizations on a 25x25x25 grid were simulated independently.
Fig. 5 Thermal conductivity: representative realizations of orientation texture $\phi(X)$ (upper row) and the corresponding temperature fields $\theta(X)$ (lower row).

Fig. 6 Thermal conductivity: data correlations. 

(a) conductivity-orientation correlation

(b) conductivity-mixture correlation

(a) average conductivity is essentially uncorrelated with the obvious feature of average orientation, 
(b) average conductivity is correlated with mixture and reciprocal mixture rules albeit with significant bias.
Figure 7 also illustrates the inhomogeneities in the stress response (at 0.3% strain in tension), which are particularly marked at grain boundaries. The 2D realizations had 5–30 grains (with a mean at 13.9) and the 3D realizations had 1–34 grains (with a mean at 13.0).

The resulting stress–strain response curves shown in Fig. 8a vary due to the particular texture $\phi(X)$ of the realization. Clearly the different crystal textures evoke different effective elastic slopes, yield points and (post-peak) flow stresses. The correlation with the initial orientation decreases with strain and increasing plasticity as Fig. 8b shows.
Fig. 9 Stress $\sigma$ for viscoelastic matrix with stiff elastic inclusions for strain $\epsilon=0.5$ at time 83.33 $\mu$s

Fig. 10 Stress: realizations of $\sigma(t)$ (where colors distinguish realizations) and correlations of the initial modulus $\partial_\epsilon \sigma(t=0)$ with the average volume fraction $\phi$.

5.3 Composites of a viscoelastic matrix and stiff inclusions

This exemplar studies another common microstructure: a composite material composed of a viscoelastic (polymer) matrix and stiff, spherical elastic (glass) inclusions. Unlike the CP exemplar these realizations employ an unstructured mesh. Another distinction from the previous exemplar is: for the matrix-inclusion composite most of the inclusions interact directly with the matrix (few interact directly with other inclusions), while in the CP exemplar grains have interfaces with a fairly uniform number of other grains. Figure 9 illustrates the variety of inclusion sizes (polydispersity) and the stress concentrations at the inclusion-matrix boundaries (plotted in reference configuration at 50% strain in tension). For this dataset 3000 3D realizations were generated with mesh sizes ranging from 23,152 to 28,892 elements (mean 25,430.9) with 16–24 inclusions (mean 17.3).

As Fig. 10a shows, the stress responses largely self-similar; however, the initial modulus plotted in Fig. 10b (nor the average stress) is not perfectly correlated with average porosity as it would be in a simple mixture. In this case $\phi(X) = 0$ for the matrix and $\phi(X) = 1$ for an inclusion, i.e. the properties of the constituents are not included in the training data.

6 Results

In this section we compare the performance of the proposed multi-level graph convolutional neural network to an analogous full graph GCNN (fGCNN), the original clustered GCNN (oGCNN) with a binary adjacency [5], and for the structured grid data analogous pixel-based CNNs. The structured grid data also allows comparison with CNN and facilitates filter visualization.
The available data described in Sect. 5 is split in a 7:2:1 ratio into training:test:validation sets. Using the training and testing data the networks were trained with the Adam stochastic optimizer [45] with learning rate 0.001 until there were 100 epochs of non-improvement in the mean squared error loss on the test data (up to 1000 epochs). A batch size of 32 was employed and all input data was normalized to $[0, 1]$ prior to training. All reported (root mean squared) errors and correlations are computed with the held-out validation data.

6.1 Performance of architecture variants

In Fig. 11 we compare the performance of variants suggested by Fig. 2 with the two large 3D datasets. The *original* architecture refers to a GCNN where the input is only the clustered data $X^*$ albeit with no attempt at featurization (unlike in Ref. [5]). Other four architectures are illustrated in Fig. 3. The *full* variant is from Ref. [9] and the other three, namely the vee (vGCNN), down (dGCNN), and reduced (rGCNN), are newly introduced in this paper. After preliminary, manual tuning the architectures for the polycrystal dataset were given $N_{\text{filters}} = 16$, $N_{\text{conv}} = 2$ (the total number of convolutional layers) and $N_{\text{dense}} = 3$ (the number of dense layers post global pooling); for the composite dataset hyperparameters were the same, except the number of filters $N_{\text{filters}} = 8$. Fully optimizing the hyperparameters was not possible since training these networks took 8–32 h on a modern GPU.

As shown in Fig. 11, all variants that incorporate data at the level of the discretization perform better than the original without manual featurization. The dGCNN variant performs the best among the new variants. In fact for both exemplar problems the order of performance for the new variants is dGCNN, rGCNN, and vGCNN. For the polycrystal exemplar the dGCNN and rGCNN variants perform on par with the full variant, while for the composite exemplar both outperform the full variant. Also apparent is the upscaling of information from cluster to mesh level that is part of the vGCNN does not have advantages for these exemplars. The fact that the multilevel variants have performance advantages over the full treatment for the composite exemplar (versus the polycrystal exemplar) may be due to the adjacency being one (matrix) to all (inclusions) which requires a more long range interaction to represent inclusion-inclusion effects (versus the contacting crystal-crystal effects). The down allows for significant mixing of the discretization- and cluster-level information; however, the reduced has the advantage of requiring considerably fewer convolutional operations which for these exemplars the number of elements versus clusters are approximately 10,000:10.
6.2 Sensitivity to the number of hidden features

A primary question regarding deep featurization, particularly with the reduced network (rGCNN), is whether the graph reduction prevents the discovery of all the features necessary to predict the output of interest. Specifically by choosing $N_{\text{filters}}$ and $N_{\text{features}}$, refer to Fig. 4, large enough can we create a rGCNN with performance on par with a full GCNN. This study shows the convergence with the number of filters and serves to set the architecture for the subsequent study. We perform two investigations, one for the 2D unstructured heat flux data and another for the 2D structured crystal plasticity data.

For the heat flux data study we first vary the number of filters that determine $N_{\text{features}}$ in the full GCNN. Figure 12a shows that the performance improves up to 2–3 features and then it remains more or less flat. Since this indicates 3 features are sufficient, for the rGCNN $N_{\text{features}}$ (the number of filters in the post-featurization convolutional stack) is fixed at 3 and $N_{\text{filters}}$ which determines the embedding is varied. For this scenario, Fig. 12a shows that $N_{\text{filters}} = N_{\text{features}}$ is sufficient to produce equivalent performance to the fGCNN. For this data the size of the output MLP is $N_{\text{dense}} = 2$, including the linear mixing layer.

For the crystal plasticity study, Fig. 12b shows the convergence of the rGCNN for $N_{\text{filters}} = N_{\text{features}}$ as well as the convergence for the corresponding fGCNN. A decrease in improvement with increasing complexity for each configuration is achieved with $N_{\text{features}} \geq 4$. It is not surprising that more hidden features are necessary to represent the more complex process. For this data only a linear mixing layer ($N_{\text{dense}} = 1$) processes the global features before they are sent to the long short-term memory (LSTM) [46] recurrent neural network (RNN) unit.
The fact that performance of these networks is tolerant to redundancy will be revisited in Sect. 6.3.

For the optimal architectures ($N_{\text{features}} = 3$ and $N_{\text{filters}} = 2$ for the heat flux example and $N_{\text{features}} = 3$ and $N_{\text{filters}} = 3$ for the stress example) Fig. 13 shows the distribution of errors for: (a) conductivity data on an unstructured mesh and (b) the stress evolution predictions using a structured mesh. The errors for the rGCNN are on par with the fGCNN for the heat flux example and slightly better for the more complex stress data. Also the rGCNN exhibits less bias than the fGCNN on the more complex data.

### 6.3 Interpretation

Using the correlation and visualization techniques described in Sect. 4 we can provide some insight into how the GCNNs are representing the response functions. Here we focus only on the CP response since we also use the CNN architecture for comparison and the response goes through different regimes (i.e. elastic-yield-plastic flow). Note the CNN has much richer kernels (9 independent parameters for a kernel width of 3, instead of 2 for the GCNN with self and first neighbor weights).

Figure 14 shows the correlation of the output of the three filters $F_a$ with the ultimate output of the network $\bar{\sigma}(t)$ as a function of applied strain $\bar{\varepsilon}(t)$. Clearly certain filters correlate with the output more strongly in certain regimes and are likely weighted more heavily in those regimes by the post-GCNN/pre-RNN mixing layer, refer to Fig. 4. The transition in importance occurs at the elastic–plastic transition for the ensemble, refer to Fig. 8a, with the low strain regime being elastic and the higher strain associated with plastic flow. Remarkably the GCNN replicates the correlations of two of the three CNN filters precisely, the third GCNN filter is not active which is likely due to the richer kernel space of the CNN not being representable in a GCNN filter [9]. The rGCNN has similar trends albeit different levels of correlation throughout the stress evolution. It appears to use its filters in a more coordinated way in order to represent the output of interest.

Regarding filter activation for the ensemble of data all filters in the CNN convolutional stack produce non-zero output, whereas for the GCNN in both the first and the second layers only two of the three are active. This is somewhat surprising but it appears that the training is producing sparsity induced by redundancy in the information the filters can produce. For the rGCNN all filters in the stack are active but appear to be producing redundant output.

The density plots in Fig. 15 show the correlation of input $\phi_i$ and filter output $Z_{ia}$ per node (or pixel). Here $i$ indexes nodes and $a$ indexes filters. Clearly there is some similarity in the features $Z_{ia}$ each network are producing. The primary filter (left column in Fig. 15) in the elastic regime folds most of the grain orientations $\phi$ through a periodic bilinear function, similar to $\cos(2\phi)$. This is consistent with expectations from analytic homogenization models, refer to Sect. 2. The primary plastic filter (middle column in Fig. 15) is more of a tent function on input values, therefore focuses on a subrange of inputs. It appears that this filter is placing the most importance on the orientations most aligned with loading. The third filter produces qualitatively different output for each architecture: for the CNN it only produces non-zero output for a different range, for the GCNN it is off, and for the rGCNN it appears to produce output loosely complemen-
Filter maps $\phi_i \rightarrow Z_{ia}$, for each pixel $i$ and filter $a$, shown as density plots.

It is also interesting that the scatter in the CNN and GCNN maps is remarkably smaller than for the rGCNN, which seems to be a signature of the relative coordination between the filters in producing the output of interest.

Lastly, Fig. 16 shows the output of each filter in the convolutional layer stack given the input on the top row. Some regions produce low values (black) while other activate high values (yellow). There is an approximate correspondence between the filter outputs of the two architectures. As discussed folding of angles to reflect the symmetry of the output and/or emphasizing of orientations most aligned with the loading create these different activations. What is not apparent in the previous analysis is there are some signatures of local edge detection, particularly in the last outputs which are effective 2-hops from the input (the middle row effects a 1-hop diffusion). Notably it appears the GCNN places less emphasis on the inter-region differences, at least for first filter (middle row of Fig. 16). Figure 17 shows the output of rGCNN stack for two trained models. The featurization output (second row) appears to emphasize edge values more than the GCNN which make sense since once the graph is reduced there is no mechanism to hold these differences on the node feature vector aggregated to regions. Also it is apparent that filters in the same level in the stack (same row in Fig. 16) are producing redundant output. Finally, the two trained models, shown in Fig. 17, with slightly different accuracies illustrate the redundancy the representation. There are similarities between the outputs of the two models but it is hard to generate a simple, plausible explanation of how they are creating...
Fig. 16  Filter visualization for a representative sample (top row: image of microstructure) through the filter stack (lower rows) for the a CNN and b GCNN architectures.

Fig. 17  Filter visualization for a representative sample (top row) through the filter stack (lower rows) for a rGCNN with RMSE=0.0247 and b rGCNN with RMSE=0.0340.
comparable representations. This redundancy hinders interpretability. Note that in both Figs. 16 and 17 the filters are displayed in arbitrary order.

7 Conclusion

In this work we presented a generalized multi-level GCNN architecture based predetermined clustering to reach a reduced graph and have shown that it can be the basis for effective models of microstructural properties and processes. The main contributions of this work are: (a) the multi-level treatment of graph data that allows for efficient transfer of information at different scales through interscale graph operators, (b) interpretation of how the proposed GNN efficiently reduces microstructural data to predictive features without manual feature engineering, and (c) demonstration on complex exemplars with qualitative different graphs (e.g. one-to-all matrix inclusions vs. neighbor-to-neighbor polycrystals). The interpretability investigation yielded surprising results regarding the simplicity of input transformations needed to achieve a predictive model, and, in particular, the specialization of filters to produce features that predict behavior in different physical regimes. Given our results, the architecture should prove to be more efficient for large graphs and deep convolutional stacks than GCNNs operating solely on graphs of the data discretization. The generality of the framework allows for more levels, more complex message passing between layers, and multiple levels of inputs than were studied in this work.

In preliminary studies deep, soft pooling such as MinCutPool and DiffPool appeared to not train well for the homogenization problem as we have posed it. This may be due the fact that the adjacencies for our problems are determined by meshes whereas the clustering is determined by the initial data essentially independently of the mesh connectivity. To enable soft pooling to work may be simply a matter of tailoring the auxiliary pooling to the homogenization task or at least ensuring the regression objectives is primary. A more hierarchical treatment may be necessary to efficiently process very large unstructured data and this will require learned pooling due the effects of convolution. This endeavor is left for future work. Pooling allows for construction of an encoder-decoder networks; however, presently effective decoders are limited [1,34].

We also speculate that we could improve performance if we could featureize edge information, e.g. misorientation of neighboring grains, in a deep manner. A filter acting on edge information, say $Z_A$ and $Z_B$ at neighboring nodes, would need to be permutationally invariant

$$w_1 Z_A + w_2 Z_B = w_1 Z_B + w_2 Z_A \Rightarrow (w_1 - w_2)Z_A + (w_2 - w_1)Z_B = 0$$

and hence $w_1 = w_2$ since $Z_A$ and $Z_B$ are arbitrary. So filters based on edge data such as $w(Z_A + Z_B)$ or $w \text{abs}(Z_A - Z_B)$ may boost accuracy. Dual graphs based on the discretization vertices instead of the cells [36], together with alternative convolutions, may prove to be a richer embedding or at least one that can augment the cell-based graphs used in this work.

Acknowledgements This material is based upon work supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research program. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-NA-0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

A Appendix: Data generating models

Details for the models and simulations that generated the three datasets described Sect. 5 are given here.

A.1 Polycrystal heat conduction

Steady heat conduction is governed by the partial differential equation

$$\nabla \cdot q = 0.$$  \hspace{1cm} (A.1)

where $q$ is the heat flux, expressed using the Fourier model

$$q = -\kappa \nabla \theta.$$  \hspace{1cm} (A.2)

In the description below the temperature field $\theta$ is normalized. In a crystal the conductivity $\kappa = \kappa(\phi(X))$ can exhibit anisotropy [47–49] and in a polycrystal each crystal has a different orientation $\phi$. The Kronecker product $\bigotimes$ of the rotation $R(\phi)$ with the conductivity tensor in the canonical orientation $\kappa = \sum_i \kappa_i e_i \otimes e_i$, gives the conductivity for a particular orientation:

$$\kappa(\phi) = R(\phi) \bigotimes \kappa = \sum_\alpha \kappa_\alpha R(\phi)e_\alpha \otimes R(\phi)e_\alpha.$$  \hspace{1cm} (A.3)

For our 2D demonstration we chose $\kappa_1, \kappa_2 = 1.0, 0.25$ W/m-K. The ratio of $\kappa_1$ and $\kappa_2$ is representative of nonmetallic crystals. For computational efficiency, we employed a simplified microstructure generation technique to create realizations of polycrystals in a square domain from Voronoi triangulation of random Latin Hypercube placements of vertices. The triangular (crystal) regions were assigned crystal orientation $\phi_K$ from a uniform random distribution $\mathcal{U}(0, \pi)$ and then meshed with triangles.
Steady state solutions were obtained with standard linear triangular finite elements. Dirichlet boundary conditions were applied to the left boundary, \( \theta = 0 \), and right boundary, \( \theta = 1 \), to establish a nominally uniform gradient. The upper and lower boundaries were given homogeneous Neumann boundary conditions. We computed the mean flux \( \bar{q} \) to obtain an effective conductivity for the sample via:

\[
\bar{\kappa} = \frac{L}{V \Delta \theta \Omega} \int_{\Omega} q(X) \, dV,
\]

which is the output of interest. Here \( L = 1 \) is the distance between the left and right boundaries, \( V = 1 \) is the area of the 2D sample, and \( \Delta \theta = 1 \) is the applied temperature difference. Input features are the orientation angle \( \phi_i \) and cell volume \( v_i \) for all elements \( i \).

### A.2 Crystal plasticity

Polycrystalline aggregates where crystal plasticity governs the individual grain response is another common target for homogenization. Here, the homogenized response of a representative sample of the grain structure characterized by \( \phi(X) \) is of interest. Crystal plasticity, like most plasticity models, is comprised of an algebraic relation mapping recoverable elastic strain to stress and a set of ordinary differential equations governing the evolution of irrecoverable plastic strain. Together, elastic and plastic strain comprise the total strain.

As in the heat flux exemplar in Sect. A.1, each sample is a collection of crystals assigned to convex sub-regions \( \Omega_K \) of a cube \( \Omega \), and the number of crystals comprising the sample vary sample to sample. DREAM3D [50] was used to create realistic 2D and 3D polycrystals’ realizations. The orientation \( \phi_K \) of each grain was characterized by 3 angles for the 3D samples, while 1 angle was sufficient for the 2D samples.

The face centered cubic (FCC) symmetry for each crystal was assumed, so that independent components of the elastic modulus tensor \( C \) were: \( C_{11}, C_{12}, C_{44} = 204.6, 137.7, 126.2 \) GPa which are representative of steel. In each crystal, plastic flow can occur on any of the (rotated) 12 FCC slip planes where the plastic velocity gradient

\[
L_p = L_p(S) = \sum_{\alpha} \gamma_\alpha(S) s_\alpha \otimes n_\alpha
\]

is determined by slip rates \( \dot{\gamma}_\alpha \), slip directions \( s_\alpha \), and slip plane normals \( n_\alpha \). The slip rate follows a power-law

\[
\dot{\gamma}_\alpha = \gamma_0 \left( \frac{t_\alpha}{g_\alpha} \right)^{m-1} \tau_\alpha,
\]

driven by the shear stress \( \tau_\alpha = s_\alpha \cdot S n_\alpha \) resolved on slip system \( \alpha \), with parameters: reference slip rate \( \gamma_0 = 1.0 \text{s}^{-1} \), rate sensitivity exponent \( m = 20 \), and the initial slip resistance \( g_\alpha = 122.0 \text{MPa} \) [58]. The slip resistance evolves according to [59,60]:

\[
\dot{g}_\alpha = (H - R_d g_\alpha) \sum_{\alpha} |\dot{\gamma}_\alpha|,
\]

with the hardening modulus \( H = 355.0 \text{MPa} \) and recovery constant \( R_d = 2.9 \).

Each sample was subject to tension at a strain rate of \( 1/\text{s} \) effected by minimal boundary conditions. The overall response of the aggregate sample reflects the anisotropy of each crystal which share boundaries and hence have kinematic constraints on their deformation.

### A.3 Viscoelastic matrix with stiff inclusions

To create realizations of a composite material, spheres of sizes drawn from a beta distribution and a uniform distribution of locations were packed into a cube. The resulting distribution of volume fractions of inclusions were representative of a SVE.

The stiff inclusion were assigned the elastic properties of glass (Young’s modulus 60 GPa and Poisson’s ratio 0.33), while the matrix was taken to be silicone, a common engineering polymeric material. We employed a Universal Polymer Model (UPM) [61,62] to model the silicone. The UPM is a viscoelastic model for the Cauchy stress \( \mathbf{T} \) of the hereditary integral type:

\[
\mathbf{T} = (K - K_\infty) \int_0^t (f_K(t-s) \dot{\epsilon}(s)) \, ds \, \mathbf{I} + K_\infty (\mathbf{tr} \dot{\epsilon}) \, \mathbf{I} + 2(G - G_\infty) \int_0^t (f_G(t-s) \dot{\epsilon}(s)) \, ds + 2G_\infty \dot{\epsilon}
\]

\[(A.9)\]
based on a bulk($K$)/shear($G$) split. The strain measure $\varepsilon$ is given by the integration of the unrotated rate of deformation $D = \frac{1}{2}(\nabla\varepsilon + \nabla\varepsilon^T)$:

$$\varepsilon = \int_0^1 R^T(s)D(s)R(s) \, ds,$$

(A.10)

where $R$ is the rotation tensor from the polar decomposition of the deformation gradient $F$ and $v$ is velocity. The relaxation kernels $f_K$ and $f_G$ are represented with Prony series with 20 relaxation times ranging from $1 \mu$s to $3160 \mu$s. The instantaneous bulk and shear moduli, $K = 920 \text{ MPa}$ and $G = 0.362 \text{ MPa}$, and equilibrium bulk and shear moduli, $K_\infty = 920 \text{ MPa}$ and $G_\infty = 0.084 \text{ MPa}$, and all other parameters are given in Ref. [62].

As with the CP realizations, each composite sample was subject to tension at a strain rate of 6000/s effected by minimal boundary conditions.

**References**

1. Hamilton William L, Ying Rex, Leskovec Jure (2017) Representation learning on graphs: methods and applications. arXiv:1709.05584
2. Bronstein Michael M, Bruna Joan, LeCun Yann, Szlam Arthur, Vandergheynst Pierre (2017) Geometric deep learning: going beyond Euclidean data. IEEE Signal Process Mag 34(4):18–42
3. Battaglia Peter W, Hamrick Jessica B, Bapst Vincent, Sanchez-Gonzalez A, Zambaldi V, Malinowski M, Tacchetti A, Raposo D, Santoro A, Faulkner R et al. (2018) Relational inductive biases, deep learning, and graph networks. arXiv:1806.01261
4. Xie Tian, Grossman Jeffrey C (2018) Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. Phys Rev Lett 120(14):145301
5. Vlassis Nikolaos N, Ma Ran, Sun WaiChing (2020) Geometric deep learning for computational mechanics part I: anisotropic hyperelasticity. Comput Methods Appl Mech Eng 371:113299
6. Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE (2017) Neural message passing for quantum chemistry. In: Proceedings of the 34th international conference on machine learning-volume 70, pp 1263–1272. JMLR. org
7. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. arXiv:1606.09375
8. Kipf Thomas N, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv:1609.02907
9. Frankel AL, Safta C, Allemann C, Jones R (2022) Mesh-based graph convolutional neural networks for modeling materials with microstructure. J Mach Learn Model Comput 3(1):1–30
10. Dominguez M, Such FP, Sah S, Pucha R (2017) Towards 3d convolutional neural networks with meshes. In: 2017 IEEE international conference on image processing (ICIP), pp 3929–3933. IEEE
11. Ogoke F, Meidani K, Hashemi A, Farinani AB (2020) Graph convolutional neural networks for body force prediction. arXiv:2012.02232
12. Pfaff T, Fortunato M, Sanchez-Gonzalez A, Battaglia PW (2020) Learning mesh-based simulation with graph networks. arXiv:2010.03409
13. Shi N, Xu J, Wurster SW, Guo H, Woodring J, Van Roekel LP, Shen H-W (2022) Gnn-surrogate: a hierarchical and adaptive graph neural network for parameter space exploration of unstructured-mesh ocean simulations. arXiv:2202.08956
14. Bianchi FM, Grattarola D, Alippi C (2020) Spectral clustering with graph neural networks for graph pooling. In: International conference on machine learning, pp 874–883. PMLR
15. Bakhvalov NS, Pansanenko G (2012) Homogenisation: averaging processes in periodic media: mathematical problems in the mechanics of composite materials, vol 36. Springer, Berlin
16. Nemat-Nasser S, Hori M (2013) Micromechanics: overall properties of heterogeneous materials. Elsevier, Amsterdam
17. Mura T (2013) Micromechanics of defects in solids. Springer, Berlin
18. Yu S, Shi J (2003) Object-specific figure-ground segregation. In: 2003 IEEE computer society conference on computer vision and pattern recognition, 2003. Proceedings., volume 2, pp II–39–IEEE
19. Karypis George, Kumar Vipin (1998) A fast and high quality multi-level scheme for partitioning irregular graphs. SIAM J Sci Comput 20(1):359–392
20. Ryu S, Lim J, Hong SH, Kim WY (2018) Deeply learning molecular structure-property relationships using attention-and gate-augmented graph convolutional network. arXiv:1805.10988
21. Kipf TN, Welling M (2016) Variational graph auto-encoders. arXiv:1611.07308
22. Nghia-Do D, Osowski S (1998) Shape recognition using FFT pre-processing and neural network. COMPEL Int J Comput Math Electron Electric Eng 17(5):658–666. https://doi.org/10.1108/03321649810221017
23. Dhillon Inderjit S, Guan Yuqiang, Kulis Brian (2007) Weighted graph cuts without eigenvectors a multilevel approach. IEEE Trans Pattern Anal Mach Intell 29(11):1944–1957
24. Ying R, You J, Morris C, Ren X, Hamilton W, Leskovec J (2018) Hierarchical graph representation learning with differentiable pooling. In: Proceedings of the 32nd international conference on neural information processing systems. Curran Associates Inc., Red Hook, NY, USA, pp 4805–4815
25. Grattarola D, Zambon D, Bianchi FM, Alippi C (2021) Understanding pooling in graph neural networks. arXiv:2110.05292
26. Samek Wojciech, Binder Alexander, Montavon Grégoire, Lapuschkin Sebastian, Müller Klaus-Robert (2016) Evaluating the visualization of what a deep neural network has learned. IEEE Trans Neural Netw Learn Syst 28(11):2660–2673
27. Qin Z, Yu F, Liu C, Chen X (2018) How convolutional neural network see the world-A survey of convolutional neural network visualization methods. arXiv:1804.11191
28. Yosinski J, Clune J, Nguyen A, Fuchs T, Lipson H (2015) Understanding neural networks through deep visualization. arXiv:1506.06579
29. Simonian K, Vedaldi A, Zisserman A (2013) Deep inside convolutional networks: visualising image classification models and saliency maps. arXiv:1312.6034
30. Bodria F, Giannotti F, Guidotti R, Naretto F, Pedreschi D, Rinzivillo S (2021) Benchmarking and survey of explanation methods for black box models. arXiv:2102.13076
31. Zhang Y, Tiño P, Leonardis A, Tang K (2021) A survey on neural network interpretability. IEEE Trans Emerg Topics Comput Intell 5(5):726–742. https://doi.org/10.1109/TETCI.2021.3100641
32. Ribeiro MT, Singh S, Guestrin C (2016) Model-agnostic interpretability of machine learning. arXiv:1606.05386
33. Ruge JW, Stüben K (1987) Algebraic multigrid. In: Multigrid methods, volume 2, pp 73–130
34. Gao H, Ji S (2019) Graph U-nets. In: International conference on image processing (ICIP), pp 3929–3933. IEEE
35. Shen H-W (2022) Gnn-surrogate: a hierarchical and adaptive graph neural network for parameter space exploration of unstructured-mesh ocean simulations. arXiv:2202.08956
36. Nemat-Nasser S, Hori M (2013) Micromechanics: overall properties of heterogeneous materials. Elsevier, Amsterdam
37. Mura T (2013) Micromechanics of defects in solids. Springer, Berlin
38. Yu S, Shi J (2003) Object-specific figure-ground segregation. In: 2003 IEEE computer society conference on computer vision and pattern recognition, 2003. Proceedings., volume 2, pp II–39–IEEE
39. Karypis George, Kumar Vipin (1998) A fast and high quality multi-level scheme for partitioning irregular graphs. SIAM J Sci Comput 20(1):359–392
40. Ryu S, Lim J, Hong SH, Kim WY (2018) Deeply learning molecular structure-property relationships using attention-and gate-augmented graph convolutional network. arXiv:1805.10988
41. Kipf TN, Welling M (2016) Variational graph auto-encoders. arXiv:1611.07308
42. Nghia-Do D, Osowski S (1998) Shape recognition using FFT pre-processing and neural network. COMPEL Int J Comput Math Electron Electric Eng 17(5):658–666. https://doi.org/10.1108/03321649810221017
36. Meyer Paul P, Bonatti Colin, Tancogne-Dejean Thomas, Mohr Dirk (2022) Graph-based metamaterials: deep learning of structure-property relations. Mater Des 223:111175
37. Xue Tianju, Adriaenssens Sigrid, Mao Sheng (2023) Learning the nonlinear dynamics of mechanical metamaterials with graph networks. Int J Mech Sci 238:107835
38. Frankel Ari L, Jones Reese E, Alleman Coleman, Templeton Jeremy A (2019) Predicting the mechanical response of oligocrystals with deep learning. Comput Mater Sci 169:109099
39. Hill Richard (1952) The elastic behaviour of a crystalline aggregate. Proc Phys Soc Sect A 65(5):349
40. Kocks UF, Tomé CN, Wenk H-R (1998) Texture and anisotropy: preferred orientations in polycrystals and their effect on materials properties. Cambridge University Press, Cambridge
41. Bianchi FM, Grattarola D, Alippi C, Livi L (2019) Graph neural networks with convolutional ARMA filters. arXiv:1901.01343
42. Jones RE, Frankel AL, Johnson KL (2021) A neural ordinary differential equation framework for modeling inelastic stress response via internal state variables. arXiv:2111.14714
44. Grattarola Daniele (2020) Spektral. https://graphneural.network
46. Hochreiter Sepp, Schmidhuber Jürgen (1997) Long short-term memory. Neural Comput 9(8):1735–1780
48. Gofryk K, Du S, Stanek CR, Lashley JC, Liu X-Y, Schulze RK, Smith JL, Safarik DJ, Byler DD, McClellan KJ et al (2014) Anisotropic thermal conductivity in synthetic diamond films. Nature 359(6394):401–403
50. Groeber MA, Jackson MA (2019) DREAM.3D. http://dream3d.bluequartz.net
51. Geoffrey Ingram Taylor (1934) The mechanism of plastic deformation of crystals. Part I. Theoretical. Proc. R. Soc. Lond. Ser. A 145(855):362-387
52. Kroner E (1961) On the plastic deformation of polycrystals. Acta Metall. 9(2):155–161
53. Bishop JFW, Hill Rodney (1951) XLVI. A theory of the plastic distortion of a polycrystalline aggregate under combined stresses. Lond. Edinb. Dublin Philos. Mag. J. Sci. 42(327):414–427
55. Mandel Jean (1965) Généralisation de la théorie de plasticité de WT Koiter. Int. J. Solids Struct. 1(3):273–295
56. Dawson Paul R (2000) Computational crystal plasticity. Int. J. Solids Struct. 37(1–2):115–130
57. Roters Franz, Eisenlohr Philip, Hantcherli Luc, Tjahjanto Denny Dharmawan, Bieler Thomas R, Raabe Dierk (2010) Overview of constitutive laws, kinematics, homogenization and multiscale methods in crystal plasticity finite-element modeling: Theory, experiments, applications. Acta Mater. 58(4):1152–1211
58. Jones RE, Templeton JA, Sanders CM, Ostien JT (2018) Machine learning models of plastic flow based on representation theory. Comput Model Eng Sci 117(3):309–342. https://doi.org/10.31614/cmes.2018.04285
59. Kocks UF (1976) Laws for work-hardening and low-temperature creep. J Eng Mater Technol 98(1):76–85
60. Mecking H, Kocks UF, Fischer H (1976) Hardening, recovery, and creep in fcc mono-and polycrystals. In: Presented at the 4th international conference on strength of metals and alloys, Nancy, 30 Aug.-3 Sep. 1976
61. Adolf Douglas B, Chambers Robert S, Neidigk Matthew A (2009) A simplified potential energy clock model for glassy polymers. Polymer 50(17):4257–4269
62. Long KN, Brown JA (2017) A linear viscoelastic model calibration of sylgard 184. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States). https://www.osti.gov/biblio/1365535

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