Journal of Machine Learning for Modeling and Computing, 3(1):1–30 (2022)

MESH-BASED GRAPH CONVOLUTIONAL NEURAL NETWORKS FOR MODELING MATERIALS WITH MICROSTRUCTURE

Ari Frankel, Cosmin Safta, Coleman Alleman, & Reese Jones*

Sandia National Laboratories, Livermore, California 94551, USA

*Address all correspondence to: Reese Jones, Sandia National Laboratories, Livermore, California 94551, USA; Tel.: +1 925 294 4744; Fax: +1 925 294 2276, E-mail: rjones@sandia.gov

Original Manuscript Submitted: 6/28/2021; Final Draft Received: 9/21/2021

Predicting the evolution of a representative sample of a material with microstructure is a fundamental problem in homogenization. In this work we propose a graph convolutional neural network that utilizes the discretized representation of the initial microstructure directly, without segmentation or clustering. Compared to feature-based and pixel-based convolutional neural network models, the proposed method has a number of advantages: (a) it is deep in that it does not require featurization but can benefit from it, (b) it has a simple implementation with standard convolutional filters and layers, (c) it works natively on unstructured and structured grid data without interpolation (unlike pixel-based convolutional neural networks), and (d) it preserves rotational invariance like other graph-based convolutional neural networks. We demonstrate the performance of the proposed network and compare it to traditional pixel-based convolution neural network models and feature-based graph convolutional neural networks on multiple large datasets.

KEY WORDS: *graph neural network, material microstructure, homogenization*

1. INTRODUCTION

Predicting the evolution of a system with a complex initial state represents a wide class of physical problems of scientific and technological interest. For instance, simulating the evolution of materials with complex microstructure is necessary for predicting the behavior of highly engineered materials (Ghosh and Dimiduk, 2011; Herriott and Spear, 2020; Kraft et al., 1996; Li et al., 2017; Stenzel et al., 2016; Yin et al., 2008). With the advent of machine learning for physical applications and the availability of considerable experimental and high-fidelity simulation data, models and architectures for these and related applications have begun to arise (Frankel et al., 2019, 2020; Pandey and Pokharel, 2020; Vlassis et al., 2020). These models can be used for a number of tasks such as subgrid accurate constitutive modeling (Frankel et al., 2019), material design by structure property exploration (Noh et al., 2019), and uncertainty quantification of materials with high intrinsic variability (Khalil et al., 2021). For this work, we are interested in predicting the evolution of the physical response of a sample given its initial state and a history of loading. For this class of problems, we assume the initial state can be represented as a field or collection of fields captured in a multispectral/multichannel image and this image is data on a structured grid or an unstructured mesh.

2689–3967/22/$35.00

In the ever-expanding field of machine learning (ML) (Bishop, 2006; Goodfellow et al., 2016; Hastie et al., 2005), there are many methods suitable to the task of supervised learning where the objective is to represent an input-output map to high fidelity. Neural networks (NN) (Goodfellow et al., 2016; Hopfield, 1982) are a particularly versatile subcategory of machine learning techniques suitable for regression tasks. They can be designed to be smooth, expressive models of physical behavior and have been shown to be effective in reducing complex processes to low-dimensional latent spaces (Fulton et al., 2019; Lee and Carlberg, 2019). In particular, convolutional neural networks (CNNs) (Albawi et al., 2017; LeCun et al., 1989; O'Shea and Nash, 2015; Qin et al., 2018) are an efficient version of fully connected networks applied to image data. Their main advantage is the reduction of the weight space needed to encode images to a manageable size by exploiting spatial correlation, locality, and similarity with a compact kernel filter. They have been spectacularly successful in a variety of image and time series applications (Krizhevsky et al., 2012; LeCun et al., 1998; Liu et al., 2017).

Graph convolutional neural networks (GCNNs) (Bronstein et al., 2017; Bruna et al., 2013) can be seen as counterparts to CNNs which operate on topologically related, as opposed to spatially or temporally related, data. Bruna et al. (2013) recognized that many of the properties that make CNNs effective on gridded data could be translated to graph-based data, such as data on unstructured meshes. They developed both local (e.g., pooling) and spectral/global (e.g., convolution in the Fourier domain) operations on graph data. They made the connection with Fourier bases through the graph Laplacian of a binary adjacency matrix to translate the convolution operation to graphs. In another arena, graph wavelets/graph signal processing, some of the graph convolution developments were preceded by Hammond et al. (2011) who developed the mathematics of spectral analysis and filtering on the more general context of kernel-weighted graphs. In order to surmount the global and expensive nature of applying filters in the spectral domain, where an eigen-decomposition is required, Defferrard et al. (2016) (ChebNet) introduced spatially compact polynomial filters. In particular, they approximated the action of a general spectral filter with a truncated Chebychev expansion of the eigenvalue matrix of the graph Laplacian, leveraging the fact that repeated application of these filters (k times) leads to k-hop diffusion of information. Kipf and Welling (2016a) took these developments to their logical conclusion with the graph convolutional network (GCN). They truncated the Chebychev expansion to first order and relied on deep/multilayer networks to build expressive representations. Appendix A provides a brief synopsis of the GCN which we base our developments on. For a more complete overview of GCNNs see reviews by Wu et al. (2020), Zhang et al. (2020), and Zhou et al. (2020). Note that related kernel-based NN methods, such as the work of Trask et al. (2020, 2019), exist and have been shown to be effective in physical applications. Also, Bronstein et al. (2017) provides an insightful perspective on applications of deep learning to non-Euclidean data, graphs, and manifolds as well as the relevant mathematics.

One of the issues with using convolutional NNs for physical problems is the need to preserve fundamental physical relationships such as frame indifference, where scalars are unchanged (invariant) to rotational changes in observer and tensorial quantities rotate in a corresponding way (equivariance) with rotation of the observation coordinate frame. The advantages of preserving these spatial symmetries in filter based NNs was recognized early on (Rowley et al., 1998). By construction CNNs embed shift/translation invariance where every local neighborhood is processed in a similar manner. Traditional convolutional filters operate strictly on structured grids of pixelated images and generally do not preserve rotational symmetries but provide a richer set of filters than methods that do. Numerous efforts have been made to endow CNNs with rotational invariance. Dieleman et al. (2016) augmented the layers of a CNN to add rotations of the image

by $\pi/4$ and inversions. Cohen and Welling (2016) outlined the mathematics whereby a convolutional network will be equivariant with respect to any group, including rotation and reflections. Worrall et al. (2017) developed filters based on circular harmonics. Chidester et al. (2018) used a discrete Fourier transform to embed rotational invariance by filter augmentation. Using concepts from abstract algebra, Kondor and Trivedi (2018) proved that convolutional structure is a necessary and sufficient condition for equivariance to the action of a compact symmetry group. This finding serves as a requirement for CNN to be equivariant. Recently, Finzi et al. (2020) provided a significant extension of Cohen and Welling's treatment of small, discrete symmetry groups to continuous (Lie) groups, e.g., rotations in three dimensions, the special orthogonal group SO(3). In contrast to pixel-based CNNs, the existing graph-based filters can operate on unstructured spatial data by use of user-defined neighbor attributions but lose some spatial information in the process. Furthermore, graph-based convolutional networks can have an inherent rotational invariance, assuming the node data is invariant/equivariant, since the representation has been lifted out of its spatial embedding.

Currently, there are many applications of pixel-based CNNs to physics and mechanics problems that have data on a structured grid. Some of the early work focused on classification and simple property prediction. Chowdhury et al. (2016) used a CNN to classify microstructures. Lubbers et al. (2017) developed a CNN-based method for inferring low-dimensional microstructure representations and used the model for microstructure generation. DeCost et al. (2017) also applied CNNs to microstructure representations and used the model to connect processing to structure. Contemporaneous with these developments, Kondo et al. (2017) used a CNN to predict ionic conductivity based on microstructure. Many publications have followed these initial applications, some have targeted evolution prediction tasks. Frankel and collaborators devised a hybrid between a CNN and recurrent NN (RNN) network to predict the elastic-plastic stress response of polycrystalline samples (Frankel et al., 2019) and have used a convolutional long short-term memory architecture (convLSTM) (Shi et al., 2015), which integrates CNN and RNN aspects into a single architecture, to predict the evolution of the stress field (Frankel et al., 2020). The latter work (Frankel et al., 2020) was based on convLSTM network of Shi et al. (2015), which was developed for atmospheric predictions and provides a framework for representing the solutions of time-dependent partial differential equations. On the other hand, we are aware of only a few applications of GCNNs to physics or mechanics to date. Vlassis et al. (2020) employed a feature-based graph neural network to model the elastic interaction of grains in polycrystalline samples using an adaptation of the CNN-RNN network in Frankel et al. (2019). Chen et al. (2020) employed a GCNN trained to sparse and unstructured diffusion data to model human tissue.

In contrast to these approaches, the proposed graph-based convolutional neural network processes the structured or unstructured microstructural images directly and in an invariant manner. The adjacency matrix, which conveys/defines neighbors and spatial information, is based on the topology of the image data itself. The formulation does not require an obvious segmentation of the microstructure, which may be occluded by noise in real data. Most importantly, it does not require featurization of the multichannel/hyperspectral image data (Bessa et al., 2017; Mozaffar et al., 2019). Nevertheless, as we will show, it can benefit from obvious features but no feature engineering is needed to obtain good accuracy. The main contributions of the work are a generalization of graph/pixel CNNs for predicting homogenized response of samples with complex microstructure and a demonstrative comparison of performance relative to existing methods.

In Section 2 we describe the physical problem, which is a homogenization of physical response suitable for subgrid/multiscale applications (Frankel et al., 2019; Jones et al., 2018). In

particular we apply the methods to homogenization of the evolution of stress of a sample volume where the internal state is determined by microstructure. In Section 3 we describe the proposed neural network architecture and relate it to traditional CNNs and feature-engineered GCNNs. In Section 4 we focus on comparing the performance of pixel-based convolutional neural network (CNN), a feature-based "reduced" graph convolutional neural network (rGCNN), and the proposed "direct" graph-based convolutional neural network (dGCNN) which operates directly on structured or unstructured image data like a CNN without the need for featurization or segmentation required by the rGCNN. To assess the performance of these models, we employ two exemplars of the homogenization problem: (a) the prediction of the stress evolution in a porous metal and (b) the prediction of the stress evolution in a polycrystalline material. With these physical problems we created four datasets using (a) an ensemble of three-dimensional (3D) realizations of the porous metal, (b) an ensemble of two-dimensional (2D) realizations of polycrystals, (c) a 3D ensemble with low variance in the grain sizes, and (d) a 3D ensemble with high variance in the grain sizes, all of which are subjected to a tension deformation process. The first dataset is used to demonstrate the efficacy of the proposed mesh-based GCNNs on unstructured meshes; the remainder are used to compare CNNs to GCNNs on structured meshes. We exploit the lower computational cost of the 2D polycrystalline dataset in a deeper exploration of variants and hyper-parameters than would be possible with the 3D data. In Section 5 we summarize the findings of exploring data and parameter efficiencies, architecture variations, and adjacency manipulation, and discuss ongoing work.

2. PHYSICAL PROBLEM

Predicting the physical response of a sample given a complex initial state is representative of a general class of problems in homogenization (Mura, 2013; Nemat-Nasser and Hori, 2013). In particular, we focus on the prediction of the evolution of the volume average of stress $\boldsymbol{\sigma}$ in a representative volume V :

$$\bar{\boldsymbol{\sigma}}(t) = \frac{1}{V} \int \boldsymbol{\sigma} \left(\boldsymbol{\epsilon}(t), \boldsymbol{\phi}(\mathbf{X}) \right) \, \mathrm{d}\mathbf{X}, \tag{1}$$

given a microstructural field $\boldsymbol{\phi}(\mathbf{X})$ observed at time $t = 0$ and a time-dependent, homogeneous loading determined by the imposed strain $\boldsymbol{\epsilon}(t)$. This class of problems is the basis for multiscale models (Trovalusci et al., 2009), material structure optimization (Le et al., 2012), and material variability and uncertainty quantification (Khalil et al., 2021). The microstructural field $\boldsymbol{\phi}(\mathbf{X})$ characterizes location-dependent inhomogeneity that influences the state of the material, where $\mathbf{X}$ is the position vector in the reference configuration of the sample. Examples of $\boldsymbol{\phi}$ include phase in multiphase composites (Bouquerel et al., 2006), elastic modulus in a material with inclusions (Roduit et al., 2009) or pores (Heckman et al., 2020), and the local defect density in a defective material (Čekada et al., 2007).

Rotational equivariance requires

$$\mathbf{Q} \boxtimes \bar{\boldsymbol{\sigma}}(t, \boldsymbol{\phi}) = \frac{1}{V} \int \boldsymbol{\sigma} \left(\mathbf{Q} \boxtimes \boldsymbol{\epsilon}(t), \mathbf{Q} \boxtimes \boldsymbol{\phi}(\mathbf{X}) \right) \, \mathrm{d}\mathbf{X}, \tag{2}$$

where $\mathbf{Q}$ is an orthogonal tensor (rotation) and $\boxtimes$ is the Kronecker product, which is defined as $\mathbf{Q} \boxtimes \mathbf{A} = \mathbf{Q}\mathbf{A}\mathbf{Q}^T = \sum_{i,j} A_{ij} \mathbf{Q}\mathbf{e}_i \otimes \mathbf{Q}\mathbf{e}_j$ for a second order tensor $\mathbf{A}$, where $\otimes$ is the tensor product. In effect this is a requirement that rotation of the inputs by $\mathbf{Q}$ must lead to a corresponding rotation of the output by $\mathbf{Q}$. It has fundamental consequences for the form that the function $\boldsymbol{\sigma}(\boldsymbol{\epsilon}, \boldsymbol{\phi})$ is allowed to take (Jones et al., 2018; Silhavy, 2013).

2.1 Exemplars

We employ two exemplars to test and demonstrate performance of the NN architectures: (a) a porous metal and (b) a polycrystalline metal. The mechanical response in both systems is complex. They undergo an elastic-to-plastic transition with loading and heterogeneous deformation due to the microstructure. For simplicity and data storage/memory considerations we focus on the primary component, $\sigma(t)$, of the volume averaged stress $\bar{\boldsymbol{\sigma}}$ for both exemplars. Each dataset was created with standard, well-documented software packages (Groeber and Jackson, 2019; Salinger et al., 2016; Stewart and Edwards, 2020).

2.1.1 Porous Plasticity

In the porous metal exemplar, $\boldsymbol{\phi}(\mathbf{X})$ is the local density field with $\boldsymbol{\phi}(\mathbf{X}) = 0$ in the pores and equal to the density of the metal elsewhere, which we normalize to one. Rizzi et al. (2018) describes a similar material model. Here aluminum was chosen as a representative material.

The metal response follows from a widely employed J_2 elastic-plastic model (Lubliner, 2008) where the stress $\mathbf{S}$ is given by a linear elastic rule:

$$\mathbf{S} = \mathbb{C} : \mathbf{E}_e. \tag{3}$$

Here ":" is a double inner product that allows the fourth order elastic modulus tensor $\mathbb{C}$ to map the elastic strain $\mathbf{E}_e$ to the stress $\mathbf{S}$. Note that $\mathbf{E}_e$ is distinct from the applied strain $\boldsymbol{\epsilon}(t)$ driving the evolution of the sample. For an isotropic material like aluminum the components of $\mathbb{C}$ reduce to

$$[\mathbb{C}]_{ijkl} = \frac{E}{(1+\nu)}\left(\frac{\nu}{(1-2\nu)}\delta_{ij}\delta_{kl} + \frac{1}{2}(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk})\right), \tag{4}$$

which depends only on Young's modulus $E = 59.2$ GPa and Poisson's ratio $\nu = 0.33$.

The plastic flow is derived from the von Mises yield condition

$$\sigma_{\text{vm}}(\mathbf{S}) - \breve{\sigma}(\epsilon_p) \leq 0, \tag{5}$$

which limits the elastic regime to a convex region in stress space and offsets the elastic strain $\mathbf{E}_e$ from the total strain. Here $\sigma_{\text{vm}} = \sqrt{(3/2)\mathbf{s}\cdot\mathbf{s}}$ is the von Mises stress where $\mathbf{s} = \mathbf{S} - \text{tr}(\mathbf{S})\mathbf{I}$ is the deviatoric part of $\mathbf{S}$, and ϵ_p is the equivalent plastic which is a measure of the accumulated plastic strain computed from the plastic velocity gradient $\mathbf{L}_p$. The yield limit $\breve{\sigma}$ is given by a Voce hardening law

$$\breve{\sigma} = Y - H\exp(-\alpha\epsilon_p), \tag{6}$$

with the following parameters: initial yield $Y = 200.0$ MPa, hardening $H = 163.6$ MPa, and saturation exponent $\alpha = 73.3$.

Realizations were created by a random placement scheme of spherical voids in the sample cube with constraints on pore overlap with other pores and the sample boundary (Brown et al., 2018). This process created unit cells with mean porosity 0.09 ± 0.03 following a beta distribution and at most 20 pores per cell. The cubic samples were on the order of 1.5 mm^3 with pore radius $\approx$ 150 μm. Pores in each of the 1121 realizations we created were explicitly meshed and resulted in unstructured discretizations with 14,640 to 101,360 elements.

Each realization was subjected to quasi-static uniaxial tension up to 20% engineering strain with minimal Dirichlet boundary conditions (no lateral constraints, uniform displacement on the ends). These simulations were performed with Sierra (Stewart and Edwards, 2020). From these

simulations we extracted microstructure $\boldsymbol{\phi}(\mathbf{X})$, applied strain $\boldsymbol{\epsilon}(t)$, and volume-averaged stress $\bar{\boldsymbol{\sigma}}(t)$ data to demonstrate the efficacy of mesh-based GCNNs in Section 4.

2.1.2 Crystal Plasticity

Our second exemplar of the homogenization problem, Eq. (1), used a crystal plasticity (CP) constitutive model where $\boldsymbol{\phi}(\mathbf{X})$ is a field of crystal orientations associated with grains. Although each grain has a relatively simple response, the collective behavior is difficult to predict without a detailed simulation since each grain influences its neighbors (Frankel et al., 2019). For this exemplar, steel was chosen as a representative material.

The response of each crystal follows an elastic-viscoplastic constitutive relation based on well-known mesoscale models (Bishop and Hill, 1951a,b; Dawson, 2000; Kroner, 1961; Mandel, 1965; Roters et al., 2010; Taylor, 1934). For the crystal elasticity, we employed the same linear stress model, Eq. (3), as in the porous metal exemplar albeit with a different elastic modulus tensor $\mathbb{C}$. In this case ferrous (face centered) cubic symmetry for $\mathbb{C}$ was assumed, and the independent components of the elastic modulus tensor $\mathbb{C}$ were those of steel: $C_{11}, C_{12}, C_{44} = \{204.6, 137.7, 126.2\}$ GPa. The overall response reflects the anisotropy of each grain; however, the response of polycrystals with random orientations $\boldsymbol{\phi}(\mathbf{X})$ was determined by the collective response, which tends to isotropy with large sample sizes. In each crystal, plastic flow

$$\mathbf{L}_p = \sum_{\alpha} \dot{\gamma}_{\alpha} \mathbf{s}_{\alpha} \otimes \mathbf{n}_{\alpha}, \tag{7}$$

can occur on any of the 12 face-centered cubic slip planes, where $\mathbf{L}_p$ is the plastic velocity gradient, $\dot{\gamma}_{\alpha}$ is the slip rate, $\mathbf{s}_{\alpha}$ is the slip direction, and $\mathbf{n}_{\alpha}$ is the slip plane normal. We employed a common power-law form for the slip rate relation,

$$\dot{\gamma}_{\alpha} = \dot{\gamma}_0 \left| \frac{\tau_{\alpha}}{g_{\alpha}} \right|^{m-1} \tau_{\alpha}, \tag{8}$$

driven by the shear stress τ_{α} resolved on slip system α. The reference slip rate was chosen to be $\dot{\gamma}_0$ = 1.0 s^{-1}, the rate sensitivity exponent was $m = 20$, the initial slip is set to zero, and the slip resistance g_{α} was given the initial value g_{α} = 122.0 MPa (Jones et al., 2018). The slip resistance evolved according to (Kocks, 1976; Mecking et al., 1976)

$$\dot{g}_{\alpha} = (H - R_d g_{\alpha}) \sum_{\alpha} |\dot{\gamma}_{\alpha}|, \tag{9}$$

where the hardening modulus was chosen to be $H = 355.0$ MPa and the recovery constant was $R_d = 2.9$. Both Eqs. (8) and (9) are integrated with standard implicit numerical integrators as part of a global equilibrium solution algorithm. See Jones et al. (2018) for additional details.

For this exemplar, we created multiple sets of $\{\boldsymbol{\phi}(\mathbf{X}), \boldsymbol{\epsilon}(t); \boldsymbol{\sigma}(t)\}$ data to train and compare the NN models described in the next section: (a) a 2D dataset consisting of 12,000 realizations (Frankel et al., 2019), (b) a 3D dataset consisting of 10,000 realizations with low variance in the grain sizes, and (c) a corresponding 10,000 realization 3D dataset with high variance in the number of grains per realization. The nominal sample length for each realization was 1 μm. The variance in the grain sizes is directly related to the variety of grain topologies in the particular ensemble; this aspect will be used in explorations described in Section 4.

Although the GCNN method can be applied to unstructured grids and complex geometries of different sizes, the CNN cannot without interpolation or some other intermediate data processing. In this exemplar we chose structured computational grids to facilitate direct comparison of pixel- and graph-based methods. The 2D dataset was computed on a 32 × 32 structured FE mesh and output over 31 time steps (max strain 0.3%); and the two 3D CP datasets used a 25 × 25 × 25 mesh and output over 51 steps (max strain 0.4%). Each polycrystal realization was subjected to quasi-static uniaxial tension at a constant engineering strain-rate of $\dot{\epsilon} = 1\ \mathrm{s}^{-1}$ with minimal Dirichlet boundary conditions. The time evolution of each system was observed over a limited number of time-steps that covered the physics of interest: the transition from elastic to full plastic flow.

Realizations of the microstructure $\boldsymbol{\phi}(\mathbf{X})$ consisted of a crystal orientation vector field that encodes the rotation of a crystal in a reference orientation to that in the polycrystal (Frankel et al., 2019). The orientation vector $\boldsymbol{\phi}$ is the unit eigenvector $\mathbf{p}$ of the rotation tensor $\mathbf{R}$, which takes $\mathbb{C}$ from a canonical orientation to that of a particular grain $\mathbf{R} \boxtimes \mathbb{C}$, scaled by the rotation angle θ around that axis

$$\boldsymbol{\phi} = \theta \mathbf{p} \ \text{ such that } \ \mathbf{R}\mathbf{p} = \mathbf{p} \text{ and } \|\mathbf{p}\| = 1, \tag{10}$$

where θ can be obtained from the nonunitary eigenvalues of $\mathbf{R}$. Refer to Frankel et al. (2019) for more details. The computational cell for each realization is a cube, which is partitioned into subregions, called grains, with distinct $\boldsymbol{\phi}(\mathbf{X})$. The subregions evoke a natural topology for a grain-based graph (Vlassis et al., 2020). All polycrystal realizations where created with Dream3D (Groeber and Jackson, 2019) using spatial correlations to obtain a reasonable number of grains and angle distribution functions that gave a uniform texture. The 2D simulations were run with Albany (Salinger et al., 2016), and the 3D simulations were run with Sierra (Stewart and Edwards, 2020). See Jones et al. (2018) and Frankel et al. (2019, 2020) for related efforts.

2.2 System Characterization and Response

In each exemplar, the nonlinear plasticity model and heterogeneous microstructure evoke a complex response to loading $\boldsymbol{\epsilon}(t)$. The local stress fields reflect internal inhomogeneities at the pores or the grain boundaries, as characterized by $\boldsymbol{\phi}(\mathbf{X})$, and display large gradients at elastic-plastic transitions. Spatial averaging to extract the system response $\bar{\boldsymbol{\sigma}}(t)$ does some smoothing of the evolution, but the range of microstructures evokes a distribution of responses. The behavior is generally similar across the ensemble of realizations, but variations are difficult to predict from simple statistics such as mean grain or pore size.

2.2.1 Porous Plasticity

Figure 1 shows the distribution of sample porosities over an ensemble of 1121 realizations. The distribution follows the target beta distribution with 0.09 mean porosity and a standard deviation of 0.03.

Due to variation in the number of voids, their size, and their placement relative to each other and inside the geometry, the response varies as the pores decrease the material stiffness and neighboring pores increase local stress concentrations. The three representative realizations shown in Fig. 2 at the final strain of 20% display significant heterogeneities in their deformation due to the dense packing of large pores. The middle realization, in particular, clearly shows a plastic localization plane due to a collection of pores leading to a weak section in the sample. The

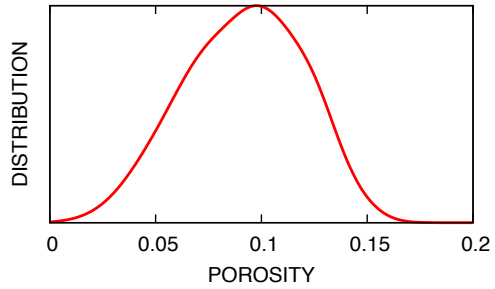


FIG. 1: Porosity distribution

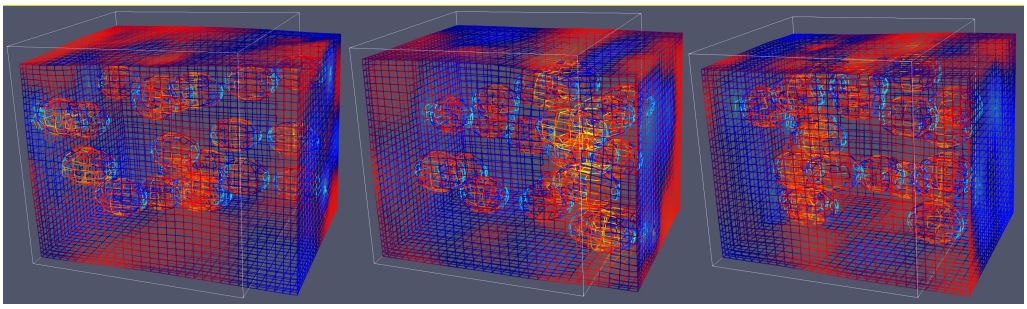


FIG. 2: Pore realizations showing exterior and interior surface mesh at 20% strain colored by tensile stress (blue: < 0, red: 700 MPa). The original, undeformed configuration is outlined.

stress field for each of the samples displays correspondingly large local variations and gradients. The average stress histories, $\{\boldsymbol{\sigma}(t)\}$, shown in Fig. 3 display a variation of 22%. After rescaling by a mixture rule based on the solid fraction, 10% variation remains. This indicates that the details of the pore configurations control a significant portion of the plastic response.

2.2.2 Crystal Plasticity

The CP microstructure also evokes a complex and evolving stress field, as Figs. 4(b)–4(d) illustrates.

Descriptive statistics of the 2D and 3D ensembles are shown in Fig. 5. The distribution of grain densities of each realization (the reciprocal of the number of grains in the particular

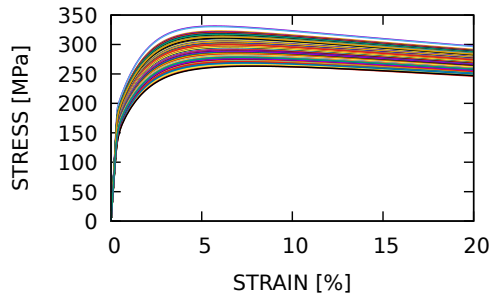


FIG. 3: Porous elastic-plastic stress response. Color distinguishes the 64 realizations shown.

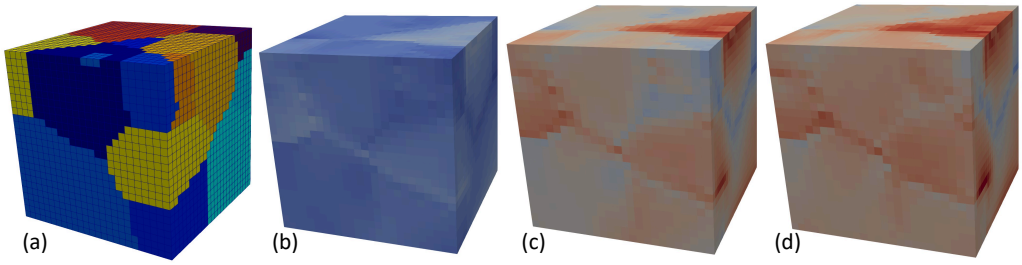


FIG. 4: Polycrystal orientation colored by (a) the first Euler angle, (b) elastic stress state, (c) transition stress state, and (d) plastic stress state, colored by σ_{11} with the same scale for all three stress state panels (blue: 0 MPa, red: 250 MPa)

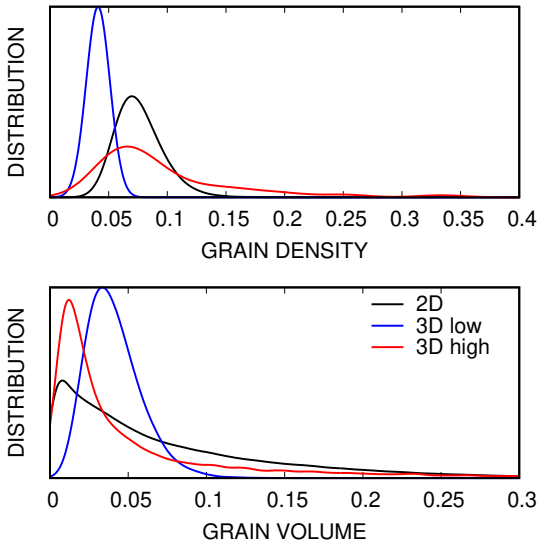


FIG. 5: Comparative statistics for 2D and 3D high- and low-variance CP ensembles. Grain densities per realization and volumes per grain have been referenced to unit cell volumes in 2D and 3D, respectively.

realization) is relatively broad and long-tailed in the high-variance 3D dataset compared to that of the low-variance 3D dataset. The grain distribution of the 2D dataset has more compact support relative to the high-varance 3D ensemble but is significantly wider than the low-varance 3D ensemble, thus representing an intermediate distribution. The distribution of the individual grain volumes over all realizations show similar trends. The peak width of the high-variance 3D dataset, however, is relatively narrow compared to the low-variance 3D dataset. The 2D data has a pronounced tail and indistinct peak which indicates a wide varance in grain sizes across the ensemble. Figure 6 illustrates how the variance of the inputs $\boldsymbol{\phi}(\mathbf{X})$ is reflected in the variance of the output $\sigma(t)$.

These datasets are particularly challenging to represent, relative to existing work, since each realization had a distinct topology/grain assignment and texture. Previous work (Frankel et al., 2019; Vlassis et al., 2020) employed a limited number of grain topologies (the tiling of the domain by distinct subregions) with unique texture assignments (the specific orientations assigned to the grains). Specifically, here each dataset had on the order of 10,000 unique topologies,

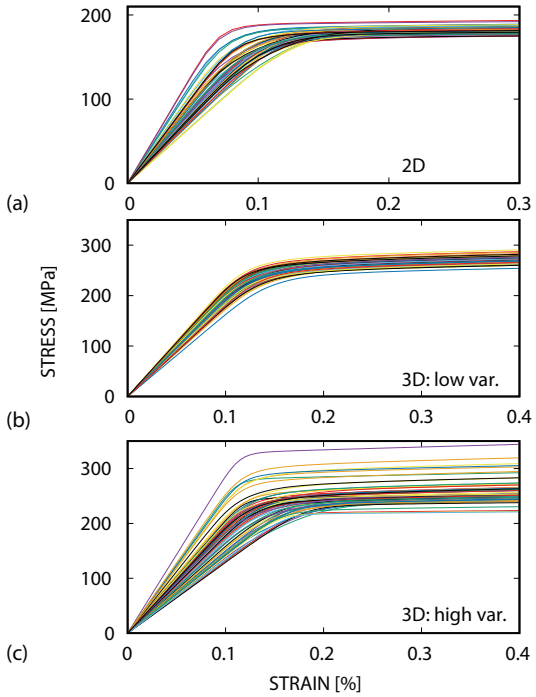


FIG. 6: CP stress response for (a) 2D, (b) 3D low grain size variance, and (c) 3D high-variance ensembles. Color distinguishes the 64 realizations shown.

whereas in Frankel et al. (2019) and Vlassis et al. (2020) there were on the order of 10–100 topologies. It is plausible, in a small dataset, when the number of convolutional filters approaches the number of microstructural topologies, a simpler learning process results since each filter can specialize to a particular topology. In this study this is not the case since each of the 10^4 samples had both a unique grain structure and grain orientation (texture). Clearly, with datasets of this size and variety, the filters must learn generalized, predictive features.

3. NEURAL NETWORK ARCHITECTURE

The overall neural network (NN) architecture to model the problem of interest Eq. (1) is analogous to the hybrid network from our previous work (Frankel et al., 2019, 2020), which was also used in Vlassis et al. (2020). It is illustrated in Fig. 7 and consists of two main components: (a) a convolutional neural network (CNN or GCNN, orange) to process the spatially complex initial state $\boldsymbol{\phi}(\mathbf{X})$ and (b) a recurrent neural network (RNN, blue) to evolve the quantity of interest $\boldsymbol{\sigma}(t)$ given the time-dependent loading $\boldsymbol{\epsilon}(t)$ as an input. As can be seen in Fig. 7, the output features of the CNN become inputs to the RNN along with the time-dependent loading. This hybrid NN has the potential to be quite complex with many hyper-parameters, e.g., a different kernel width for each convolution layer and different number of nodes for each dense layer.

In this work we simplify the NN architecture from that we previously employed to facilitate exploration and comparison of CNN and GCNN approaches. This is not particularly constraining since there are redundancies in the approximation power of such a network. We determine the shape of the network diagrammed in Fig. 7 with three hyper-parameters: (a) the number of

filters, N_f, applied in parallel to node data; (b) the number of convolutional layers, N_c; and (c) the number of densely connected layers, N_d, post-convolution used in the reduction of image to (hidden) features. In the following we will use the abbreviation $(N_\mathrm{f}{:}N_\mathrm{c}{:}N_\mathrm{d})$ to refer to architectures defined by these three parameters, for instance $(4{:}2{:}1)$ is a network with 4 filters, 2 convolutional layers, and 1 dense layer. Note the global (average) pooling after the convolutional layers reduces the output of the image processing CNN to N_f outputs and hence determines the width of the entire convolutional component including the densely connected layers. In our previous work (Frankel et al., 2019) we used an encoder for this task. Figure 8 illustrates the details of a typical convolutional component in the overall architecture (yellow in Fig. 7). A batch normalization layer (not shown) is inserted between each convolutional layer to aid in conditioning the output and training (Bjorck et al., 2018). To facilitate comparison of the CNN and GCNN approaches we fix the kernel width of pixel convolutional layer to 3 to make an analog of a graph where connections are nearest neighbors for the physical problem. The particular RNN we employ is the well-known long-short-term memory unit (LSTM) (Hochreiter and Schmidhuber, 1997). In preliminary studies we also tried another standard RNN, the gated recurrent unit (GRU) (Cho et al., 2014), which achieved marginally worse accuracy on average. The RNN used $\tanh$ activations and all other layers employ rectifying linear units (ReLUs) for their nonlinear activation functions. We also tried $\tanh$ activations for the entire network, but that configuration choice performed relatively poorly (Glorot and Bengio, 2010; Glorot et al., 2011). Lastly we apply the standard technique of using a linear mixing layer (no nonlinearity, only affine transformation) just prior to the output of interest.

The primary variation in the networks we explore is in the convolutional unit (yellow in the schematic Fig. 7 and shown in detail in Fig. 8) and in particular the construction of the convolution filter, which is our focus.

3.1 Proposed Graph Structure

At the level of digitized data in a pixel-based CNN, the microstructural input is values of $\boldsymbol{\phi}$ at pixels (or a cell or an element) that are addressed in a grid-wise fashion. In general the field $\boldsymbol{\phi}(\mathbf{X})$ can be represented as a pixelized image on a structured grid or on an unstructured mesh. For a graph representation, such as that used in Vlassis et al. (2020), the reduction of a clearly segmentable microstructure, such as that illustrated in Fig. 4(a), leads to nodes representing homogeneous regions and graph edges encoding adjacency between regions. This reduction loses spatial information such as the shape of the regions in the clustering/aggregation to nodes. Hence, that framework typically requires enrichment of the node data x by featurization, i.e., picking measures/statistics that quantify the information lost in the reduction. In that approach, the number of nodes N_{nodes} is variable and equal to the number of regions in the particular sample. In that context, and in general, the node data x is number of nodes N_{nodes} by number of input features N_f derived from $\boldsymbol{\phi}(\mathbf{X})$. Some features are obvious, such as the value of $\boldsymbol{\phi}$ in the clustered region represented by the node and the volume of the region, and others are not. It is easy to see that these features can result from preconceived filters and clustering operations applied to image $\boldsymbol{\phi}(\mathbf{X})$. We will refer to this feature-based, reduced graph convolutional network as a rGCNN.

To avoid the nebulous task of featurization, we propose a direct graph CNN (dGCNN) where x is identically $\boldsymbol{\phi}(\mathbf{X})$ at the cell/element centers of the computational grid but flattened (in an arbitrary order) to fit in the graph convolution paradigm. The graph convolutions are permutationally invariant, so ordering does not affect output. In the proposed network N_{nodes} is the number of pixels or unstructured elements in the image, and the edges are derived directly from

the mesh topology, i.e., element/pixel neighbors are graph neighbors. This approach has qualitative advantage of being a graph-based representation, which has intrinsic invariance properties, while working directly on the structured or unstructured data $\boldsymbol{\phi}(\mathbf{X})$, not a segmented or clustered version of it. This representation does not preclude the use of derived, informative features to boost accuracy by adding them to x, as we will demonstrate.

3.2 Comparison of Pixel- and Graph-Based Convolution

To understand the difference between a pixel-based and a graph-based convolution, let us examine a single convolutional filter. Note that the proposed architecture shown in Fig. 7 and Fig. 8 has several filters. In general, a convolutional filter has trainable weights W and bias b. The weight matrix W effects an affine transform by matrix multiplication that mixes input features x into output features y, and b provides an offset to tune activation of the subsequently applied nonlinearity $\mathsf{y} = f(\mathsf{x})$. In a traditional grid/pixel convolution (CNN), the node features are addressable by grid index, for instance in 2D:

$$\mathsf{y}_{(i,j)} = \sum_{k',l'} \mathsf{W}_{(k',l')} \mathsf{x}_{(i+k',j+l')} + b, \tag{11}$$

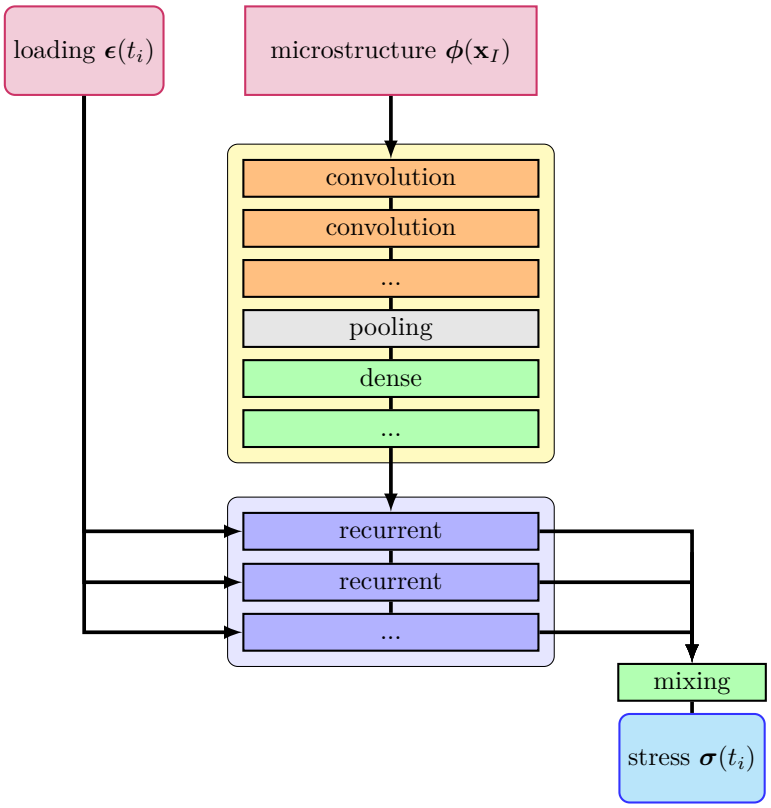


FIG. 7: Hybrid neural network with CNN (yellow/lighter) and RNN (blue/darker) components to transform the inputs (red): spatially dependent $\boldsymbol{\phi}(\mathbf{X})$ microstructure and time-dependent loading history $\boldsymbol{\epsilon}(t)$, to the output $\boldsymbol{\sigma}(\boldsymbol{\phi}, t)$ (cyan/bottom)

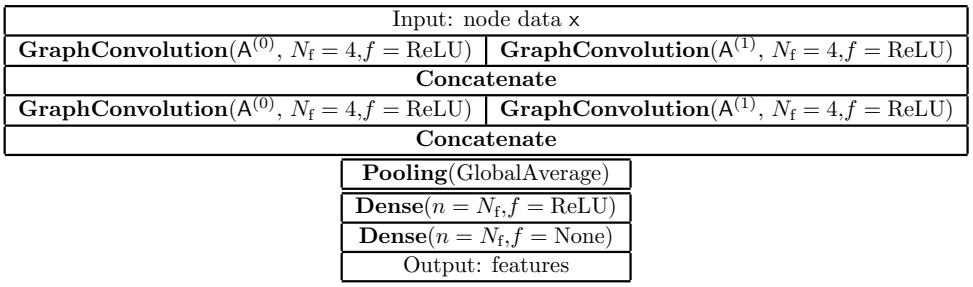


FIG. 8: A (4:2:2) convolutional unit. $\mathsf{A}^{(i)}$ is the adjacency matrix for the ith neighbors (0 is self), N_f is the number of filters, and f is the activation function for the layer. Note the last dense layer is linear (no nonlinear activation function).

where i, j are indices over the pixelated image/discretized field, and k', l' are indices over the convolution/filter which is $N_\text{kernel} \times N_\text{kernel}$ in size, with $N_\text{kernel} \ll N_\text{nodes}$ being the kernel width. To see the similarities with graph convolution we recast this convolutional multiplication in Eq. (11) by mapping both (i, j) and (k', l') to single indices I and L by imposing a fixed, arbitrary ordering across the kernel W (refer to Fig. 9). For the flattened input x, the corresponding output y is

$$\mathsf{y}_I = \sum_J \left[\sum_{K'} \mathsf{W}_{K'} \mathsf{A}_{IJ}^{(K')} \right] \mathsf{x}_J + \mathsf{b}_{K'}, \tag{12}$$

where multiplication by a matrix $\mathsf{A}^{(K')}$ provides a masking operation translating the global indices to local dependencies. Here $\mathsf{A}^{(K')}$ is a global $N_\text{nodes} \times N_\text{nodes}$ adjacency matrix for each entry K' in the convolutional kernel (i.e., each pixel under the filter kernel is treated uniquely); and $\mathsf{A}_{IJ}^{(K')} = 1$ if I and J are neighbors (by some definition, e.g., shared face, shared node in the computational mesh or distance) and 0 otherwise. The definition of neighbors determines the direct interactions, in rough analogy to choosing the kernel width in a pixel-based convolutional filter.

Likewise, for a graph convolutional network (GCN) layer (Kipf and Welling, 2016a), the convolution operation takes the form:

$$\mathsf{y}_I = \sum_J W \mathsf{A}_{IJ} \mathsf{x}_J + b, \tag{13}$$

where the adjacency A plays the same masking/connectivity role as $\mathsf{A}^{(K')}$ and an ordering of the input data x is necessary to associate the indices I and J with particular cells. Again $\mathsf{A}_{IJ} = 1$ if I and J are neighbors by some definition and 0 otherwise. Based on the derivation of the GCN (refer to Appendix A), self-loops are added such that $\mathsf{A}_{II} = 1$. For each filter there are two trainable parameters, W and b, versus k^d for a pixel-based CNN filter, where d is the spatial dimension and k is the kernel width. These graph-based filters have permutational invariance by construction since all neighbors have the same weight. They are also typically normalized by the number of neighbors, for instance in a GCN, the binary adjacency matrix $\tilde{\mathsf{A}}$ is normalized by the degree matrix $\mathsf{D}_{IJ} = \sum_I \tilde{\mathsf{A}}_{IJ} \delta_{IJ}$

$$\mathsf{A} = \mathsf{D}^{-1/2} \tilde{\mathsf{A}} \mathsf{D}^{-1/2}, \tag{14}$$

to convey average neighborhood information and improve the conditioning of A.

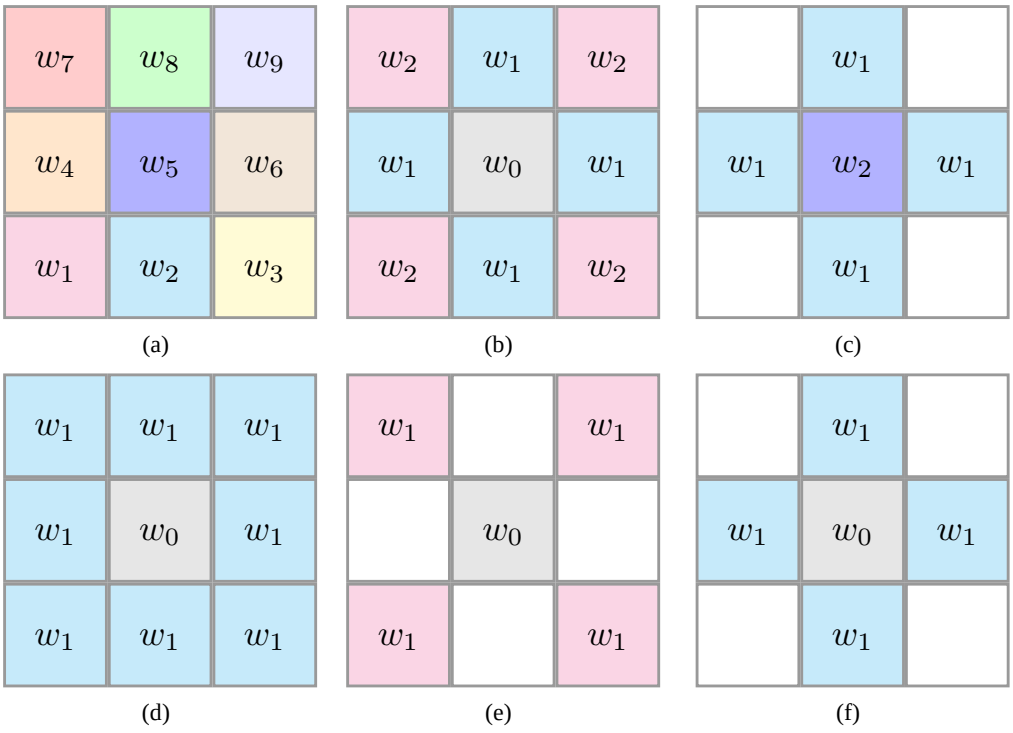


FIG. 9: Convolutional filters in 2D: (a) pixel: CNN where all neighbors have independent weights (denoted by colors); and alternative GCN-like filters that preserve invariant outputs: (b) $*$: edges and nodes have separate weights; (c) #: all edges have a single weight; (d) O: all neighbors have a single weight; (e) X: all node neighbors have single weight; (f) +: all edges have a single weight. Edge neighbors share an edge/face, whereas node neighbors only share a vertex/node. Color: trainable, gray: determined by trainable, white: not used.

Finally, in both the pixel and graph convolutional cases an activation function f is applied element-wise to the resulting node data y

$$\mathsf{x}^\dagger = f\left(\mathsf{y}(\mathsf{x})\right), \tag{15}$$

to obtain the input $\mathsf{x}^\dagger$ to the next layer from that of the present layer x.

Clearly the adjacency A plays a crucial role in convolution by defining what constitutes influence between nodes. In physical problems typically influence is local and decays with distance. For some applications, such as electrostatic interactions, the interactions do not decay quickly with distance and hence are long-ranged (Li et al., 2020). Most other applications have relatively short-ranged influence, such as contact/interface interactions. As mentioned, the source field for the microstructure is a pixelated image grid or computational mesh which has its own obvious topology and neighbors. In the proposed direct graph we define adjacency by the pixel/cell neighbors of the image grid/mesh, not by the data $\boldsymbol{\phi}(\mathbf{X})$ on it. When reduced to a graph in this fashion, where nodes are the pixels/cells, the only topologically distinct nearest neighbors are (a) those that share an edge with the particular cell or (b) only a vertex. In CNNs compact kernels are preferable since they limit the number of parameters that need to be trained. For GCNNs,

and in particular the GCN, neighbors are not distinct and hence there is only a single weight. As in CNNs, longer range influence can be captured by multiple layers applied in sequence.

Kipf and Welling (2016a) GCN is well-known to be an effective (nonspectral) convolutional filter. By comparing it to standard CNNs we employed in our previous work (Frankel et al., 2019) we devised a few variants based on manipulating the node adjacencies and associated weights. Figure 9 shows the variety of rotationally and permutationally invariant filters we explored. Note these filters assume that the neighbors are at equivalent distances from the node at the center and the cells are of comparable sizes. We generalize this type of filter to multiple adjacencies $\mathsf{A}^{(K')}$ with their associated weights $\mathsf{W}_{K'}$ via:

$$\mathsf{y}_I = \sum_J \left[\sum_{K'} \mathsf{W}_{K'} \mathsf{A}_{IJ}^{(K')} \right] \mathsf{x}_J. \tag{16}$$

The pixel-based filter in a standard CNN treats every pixel in the kernel independently [refer to Fig. 9(a)] and retains a sense of how pixels located relative to the central pixel. This richness has the side effect of not satisfying the symmetries required by invariance. To elaborate with an illustration, if the data $\{\boldsymbol{\phi}, \boldsymbol{\epsilon}; \boldsymbol{\sigma}\}$ is rotated by $\pi/2$ around a coordinate axis, the data at pixels are uniquely mapped by Q to new indices without interpolation $\mathsf{x}^* = \mathsf{Q}\mathsf{x}$. In this simple case the linear transformation

$$\mathsf{Q}\mathsf{y}_I \equiv \mathsf{Q} \sum_J \left[\sum_{K'} \mathsf{W}_{K'} \mathsf{A}_{IJ}^{(K')} \right] \mathsf{x}_J \neq \sum_J \left[\sum_{K'} \mathsf{W}_{K'} \mathsf{A}_{IJ}^{(K')} \right]^* \mathsf{Q}\mathsf{x}_J, \tag{17}$$

in Eq. (12) applied to the inputs $\mathsf{x}_J = \boldsymbol{\phi}(\mathbf{X}_J)$ will not produce a rotated output since image $\mathsf{x} = \boldsymbol{\phi}(\mathbf{X})$ rotates but the filter $[\mathsf{W}_{K'}\mathsf{A}^{(K')}]$ for the K'th adjacency does not. A graph treatment, unlike a pixel convolution, effectively allows the transformation Q to commute with the adjacency, $\mathsf{Q}\mathsf{A}^{(K')}\mathsf{x} = [\mathsf{A}^{(K')}]^*\mathsf{Q}\mathsf{x}$, since the node association of the graph adjacency is invariant to spatial transformations. Refer to Kondor and Trivedi (2018) and Finzi et al. (2020) for a more formal treatment. The equivariance problem remains for the action of the weights $\mathsf{W}_{K'}$.

Permutational invariance of the weights is one means of solving this issue. Strictly speaking, the weights of all of the neighbors of the central pixel are required to be identical for permutational invariance (and for invariance with respect to arbitrary rotations). We implemented filter variants where edge and vertex weights are used exclusively [Fig. 9(f) uses edge neighbors whereas Fig. 9(e) employs only node neighbors] or given independent weights, such as the "$*$" pattern in Fig. 9(b). These filters are in contrast to the CNN filter, shown in Fig. 9(a), which has no inherent symmetries. Note there is some evidence that with sufficient data CNNs learn rotational invariance (Quiroga et al., 2018) but the benefits of a smaller parameter space and a compact representation that satisfies this constraint exactly are clear. In the GCN, as designed by Kipf and Welling (2016a), the weight of the center [shown in gray in Fig. 9(f) vs. colored in Fig. 9(c)] was chosen to be the same as the neighbors (refer to Appendix A). In this work we also tried variants where the self-weight is independent of the neighbor weight [Fig. 9(c) vs. Fig. 9(f)]. As in Eq. (12), each independent, trainable weight $\mathsf{W}_{K'}$ can be associated with a different predetermined, binary adjacency matrix $\mathsf{A}^{(K')}$, or, equivalently, the weighted adjacency $\sum_{K'} \mathsf{A}_{IJ}^{(K')} \mathsf{W}_{K'}$ can be considered the trainable entity.

4. RESULTS

To establish the efficacy of the proposed GCNN-RNN architecture applied directly to unstructured mesh data without featurization we demonstrate its performance on the porous material dataset. Since hyper-parameter optimization and other computationally intensive tasks are more feasible with the 2D dataset, we use it explore a variety of hyper-parameters and architecture choices with the 2D dataset. After determining what graph-based filters perform well and have good accuracy per parameter and dataset size, we demonstrate the proposed architecture on the two 3D CP datasets.

The data was conditioned to aid training. The input data $\boldsymbol{\phi}(\mathbf{X})$ and $\boldsymbol{\epsilon}(t)$ were normalized by their maximum values since both had lower bounds of zero. The output data $\sigma(\boldsymbol{\phi}_a, t)$ was transformed to the difference $\Delta\sigma(\boldsymbol{\phi}_a, t)$ between the data $\sigma(\boldsymbol{\phi}_a, t)$ and its mean trend $\langle\sigma\rangle = (1/N)\sum_a \sigma(\boldsymbol{\phi}_a, t)$ (N is the number of realizations) and normalized by the standard deviation of $\sigma(\boldsymbol{\phi}_a, t)$ over time t. We chose to train the model response $\hat{\sigma}(\boldsymbol{\phi}_a, t)$ to the difference from the mean trend to emphasize the variation in response between microstructures $\boldsymbol{\phi}_a$. We evaluated the performance of each of the convolutional networks primarily with the root mean squared error (RMSE)

$$\varepsilon(t) = \frac{1}{\max\langle\sigma\rangle}\sqrt{\sum_a (\hat{\sigma}(\boldsymbol{\phi}_a, t) - \sigma(\boldsymbol{\phi}_a, t))^2}, \tag{18}$$

relative to the maximum σ over the dataset, and the (Pearson) correlation coefficient

$$C(t) = \frac{\sum_a \Delta\hat{\sigma}(\boldsymbol{\phi}_a, t)\,\Delta\sigma(\boldsymbol{\phi}_a, t)}{\sqrt{\sum_a \Delta\hat{\sigma}(\boldsymbol{\phi}_a, t)^2 \sum_a \Delta\sigma(\boldsymbol{\phi}_a, t)^2}}, \tag{19}$$

of the normalized data. In Eqs. (18) and (19) the sum is over all realizations a in the test set, and the NN model response is denoted as $\hat{\sigma}(\boldsymbol{\phi}_a, t)$. We used a 70/10/20 train/validation/test split for the smaller porous material dataset and a 80/10/10 split for the larger CP datasets.

Recall the abbreviated architecture naming convention $(N_\mathrm{f}{:}N_\mathrm{c}{:}N_\mathrm{d})$, which will be used throughout this section.

4.1 Demonstration on Unstructured Mesh Data

We used the smaller 3D porous material dataset to demonstrate that GCNNs applied directly to unstructured mesh data are effective at representing homogenized material behavior. For this study we used a (32:4:2) convolutional unit with "O" type filters that treat all nearest neighbor elements equally. The field data $\boldsymbol{\phi}(\mathbf{X})$, in this case, is the binary density field (0: void, 1: metal) on the native unstructured computational mesh. Since element and systems sizes varied across the ensemble, we augmented $\boldsymbol{\phi}(\mathbf{X})$ with the volumes. The stress response $\bar{\sigma}(t)$ is composed of 400 tensile loading steps to reach 20% strain. Since each realization has a different mesh and each adjacency matrix is large (on the order of 10^5–10^6 rows and columns) albeit sparse, we trained the network by evaluating and updating the network weights one sample at a time, i.e., a batch size of 1.

The predictions for eight randomly selected trajectories shown in Fig. 10 are smooth and display minimal errors that are fairly uniform across the evolution. The initial elastic regime is well captured, as is the ultimate (peak) strength and subsequent plastic flow. Over the entire test set of 224 samples, the normalized mean RMSE was 0.00409 with 0.00052 standard deviation, and the mean correlation was 0.995. Clearly the architecture shown in Fig. 7 with graph convolution layers applied to unstructured field data is an effective model of this microstructural response.

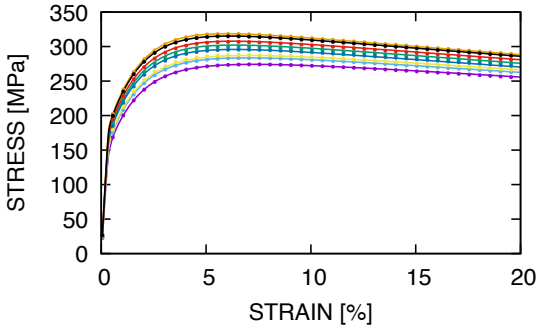


FIG. 10: Comparison of true (lines) and predictions (points with corresponding color) for eight realizations of the porous metal data

4.2 Efficacy of the Components of Hybrid Network

In this study we vary the three chosen architecture hyper-parameters: N_{f}, N_{c}, and N_{d} for three architectures: (a) a CNN, (b) the proposed dGCNN, and (c) a rGCNN endowed with angle and volume features. Both GCNN architectures employ the standard GCN filter, with a "+" pattern and dependent center weight illustrated in Fig. 9(f). All models are trained to the 2D CP dataset.

Figure 11 shows the correlations $C(t)$ and errors $\boldsymbol{\varepsilon}(t)$ over time for the three architectures for a range of filters $N_{\mathrm{f}} \in \{1, \ldots, 6\}$ for one or two convolutional layers ($N_{\mathrm{c}} = \{1, 2\}$) and one dense layer ($N_{\mathrm{d}} = 1$). Note that two dense layers, $N_{\mathrm{d}} = 2$, produced similar results. Generally the correlation of all three hybrid CNN-RNNs is better earlier in the process, while the error tends to peak early on near the end of the elastic regime where the response variance is highest. Referring to Fig. 6(a), we observe that elastic-to-plastic transition occurs around 0.1%, and this transition is apparent in Fig. 11 where the correlation appears to transition between two plateaus. All architectures improve with more filters although there is a clearly a limit to the improvement, which suggests a small number of relevant features. It is also apparent that more filters are needed to capture the later plastic regime accurately than the initial elastic regime.

The simplest CNN models (fewest parameters) that have the best performance are: $(4{:}1{:}1)$ with 269 parameters, $(3{:}2{:}1)$ with 271 parameters (shown in Fig. 11), and $(3{:}1{:}2)$ with 187 parameters, $(2{:}2{:}2)$ with 151 (not shown). This demonstrates the fungibility of the nodes in the network. The proposed dGCNN with $N_{\mathrm{c}} = 1$ with either $N_{\mathrm{d}} = \{1, 2\}$ does well in the elastic regime for $N_{\mathrm{f}} > 3$ but achieves no greater than 0.7 correlation for plastic for all cases with $N_{\mathrm{f}} < 9$; however, with a second convolutional layer three filters are sufficient to achieve the accuracy of the best CNN architectures. This indicates that second nearest neighbors are required to represent the plastic flow well using the standard GCN filter in the dGCNN architecture. This finding will be revisited in Section 4.3. Lastly, all the variants of rGCNN with angle and volume node features gave similar (poor) performance. The rGCNN clearly requires more features for improvement. This finding will be expanded on in Section 4.4.

Table 1 compares the number of trainable parameters for the three types of convolutional networks. Clearly, the additional weights in the pixel-based convolutional layers incur a cost in complexity and training. Also, the parameter complexity of the dGCNN is essentially equivalent to the rGCNN since the same filters are being used on different graphs and data. The data is certainly different, with the rGCNN storing more features on a more compact adjacency than the dGCNN; with sparse storage this is not a significant advantage for moderately sized meshes.

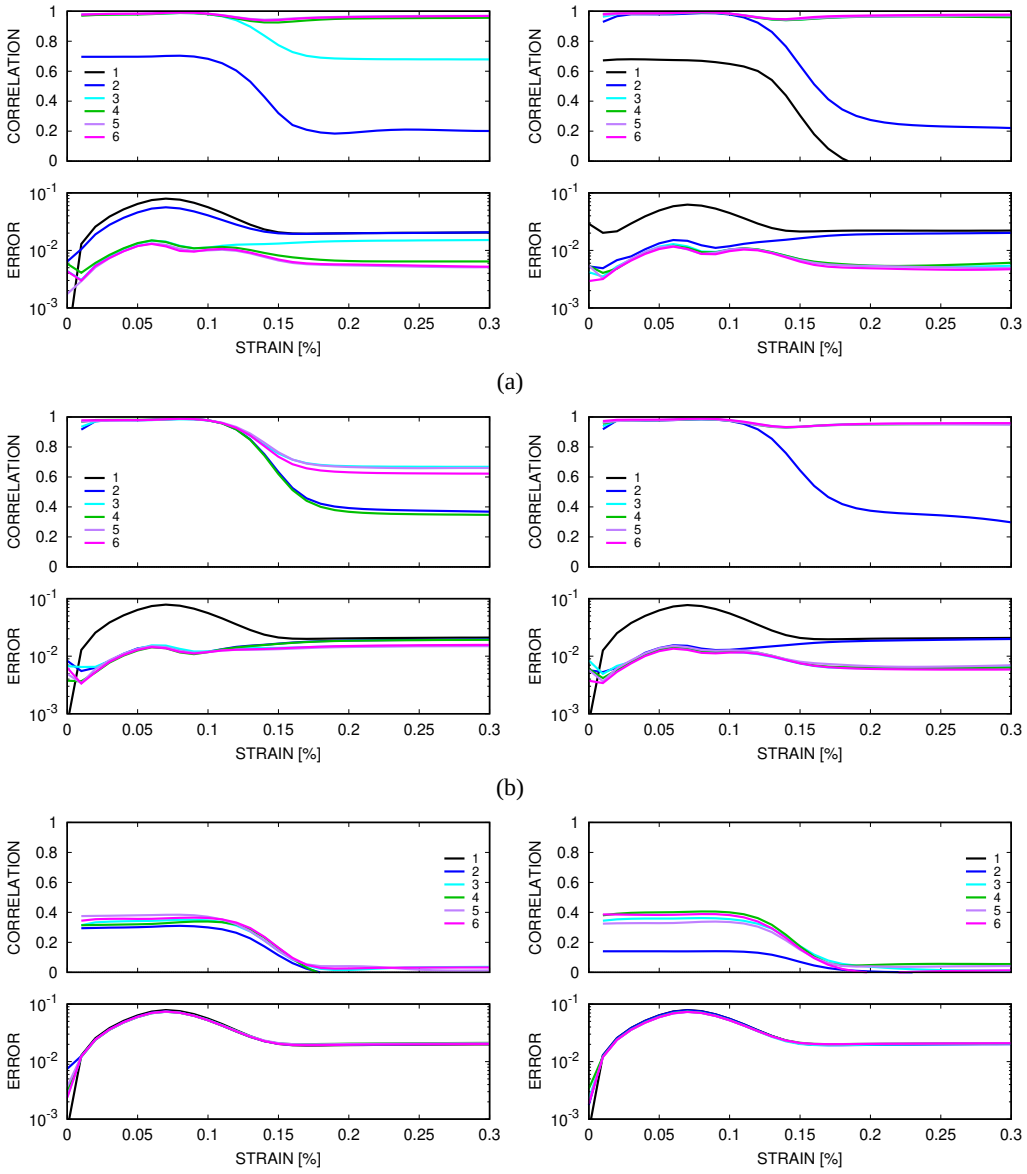


FIG. 11: Architecture comparison (2D CP data): (a) CNN, (b) dGCNN: GCN applied directly to the grid, (c) rGCNN: GCN applied to clustered/segmented data. Left panels: 1 convolutional layer, 1 dense layer; right panels: 2 convolutional layers, 1 dense layer.

4.3 Comparison of Convolutional Filters

Motivated by the fact that a CNN can achieve good performance with only one convolutional layer we tried richer variants of the standard GCN filter (where the self-weight is set equal to the neighbor weight). Using the patterns shown in Fig. 9, we explored their relative performance

TABLE 1: Parameter counts for architectures with GCN filters applied to 2D CP data

Architecture	CNN	dGCNN	rGCNN
(1:1:1)	41	26	27
(2:1:1)	99	75	71
(4:1:1)	269	209	213
(6:1:1)	511	421	427
(1:2:1)	55	32	33
(2:2:1)	145	69	85
(4:2:1)	433	245	249
(6:2:1)	865	487	493

with one convolutional layer ($N_c = 1$) and four filters ($N_f = 4$). Figure 12(a) shows that patterns +, X, O, which only have one trainable weight and have interactions with edge, vertex-only, and edge+vertex neighbors, respectively, have comparable and less than satisfactory performance. Even the $*$ pattern, where edge and node neighbors are given separate weights (two independent weights), has an inferior performance to the CNN (which has nine independent weights). Only the # pattern, where the center pixel is given an independent weight from the vertex neighbors, has performance on par with the CNN. Given this finding we endowed each of the basic patterns $\{+, X, O, *\}$ with an independent central weight. As shown in Fig. 12(b), this was sufficient to improve the performance of all but the X (vertex-only neighbors) to be comparable with the CNN. It is physically plausible that that edge neighbors have a stronger influence than vertex neighbors and it appears that they are required for the filters to learn predictive interactions.

4.4 Selected Features

In Section 4.2 we observed that the feature-dependent rGCNN formulation had subpar performance especially in the plastic regime that was not improved with a more complex GCNN component. Figure 13 shows the performance of a (4:2:1) rGCNN network does improve with an

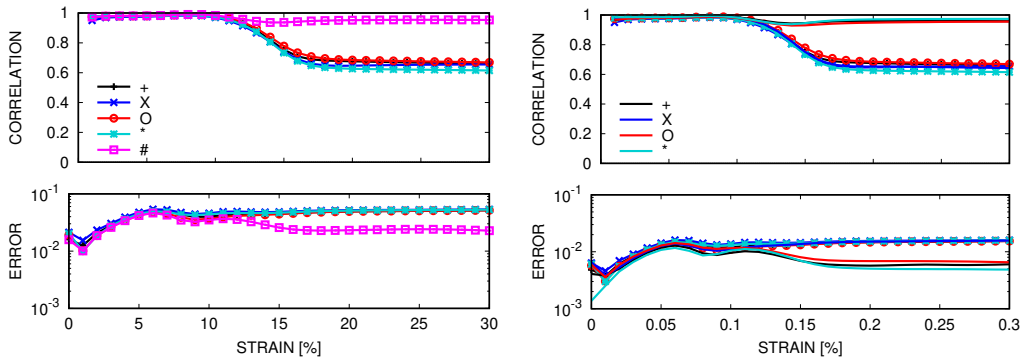


FIG. 12: Filter comparison (2D CP data): (a) patterns illustrated Fig. 9 and (b) edge, vertex, and both filters augmented with an independent center weight

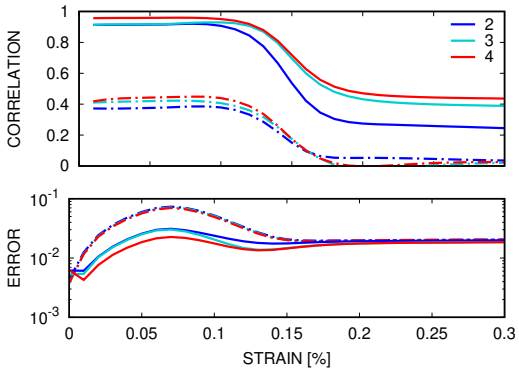


FIG. 13: Comparison of rGCNN with increasing number of features (2D CP data). Features: angle, volume, area, surface area. Graph labels correspond to 2: {angle, volume}, 3: {angle, volume, area}, and 4: {angle, volume, area, surface area}. Dashed lines: self-weight equal to neighbor weight; solid lines: self-weight independent of neighbor weight.

expanded feature set. Here, in addition to the orientation angle and the volume associated with the clusters/grains represented by the graph nodes (two total features), we added the surface area of each grain (three total features) and the area of the grain that is on the surface of the cell (four total features) to the node features. The improvement with these additional features is marginal. However, if we allow for an independent self-weight by changing the standard GCN pattern to the # pattern, as in the previous section, the performance is dramatically improved and the improvement with additional features is more distinct. Although these networks have considerably smaller adjacency matrices than dGCNN due to the clustering of the pixels, the performance is subpar, particularly in the plastic regime. This serves as an illustration of the difficulty of improvement by feature selection, as opposed to deep learning. Note only elastic response was modeled in Vlassis et al. (2020).

4.5 Data Efficiency

As a last trial with the 2D CP dataset, we investigated how efficient the best networks are with smaller datasets. Here we compared (a) $(4{:}1{:}1)$ CNN with 269 parameters, (b) $(4{:}2{:}1)$ dGCNN with the + pattern (GCNN+) and 317 parameters, and (c) $(4{:}1{:}1)$ dGCNN with the # pattern (GCNN#) and 249 parameters. The training set was reduced from 80% of the 12,000 realizations (9600) by a fraction that ranged from 0.01 to 1.0. The test set was a fixed 20% (2400) of the full realizations and the results were averaged over nine trials. Figure 14 shows that the majority of learning (improvement in accuracy) occurs by the time the training size is approximately equal to the number of parameters. After the step-down in error (at 0.04 of the total training set for the CNN, at 0.02 for the GCNN+, and at 0.05 GCNN#) the improvement is relatively slow but steady. This data demonstrates that these small networks can be effective at the prediction of the homogenized response task, Eq. (1), with much smaller datasets.

4.6 Boosting with Preconceived Features

Now that we have discovered effective adjacencies and guidance on architecture hyper-parameters, we turn to using the 3D CP datasets. Motivated by the fact that some features of the

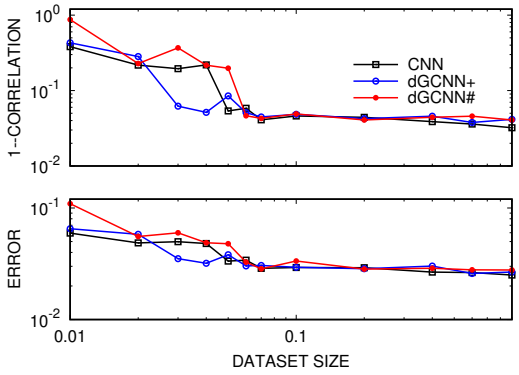


FIG. 14: 2D CP data: data efficiency comparison for (4:1:1) CNN with 269 parameters, (4:2:1) dGCNN+ 317 parameters, and (4:1:1) dGCNN# with 249 parameters. Note: 1–$C(t)$ is being plotted in the upper panel.

image $\boldsymbol{\phi}(\mathbf{X})$ have obvious bearing on the output due to physical reasoning, we boosted the dGCNN with some of the features we employed with the solely feature-based rGCNN. The proposed architecture can accommodate preselected features by simply augmenting the image/cell microstructural field $\boldsymbol{\phi}$ with additional channels. Figure 15 shows the effect of adding a channel with the volume fraction of the associated grain to each pixel. Clearly there is a distinct and uniform benefit; however, it is somewhat marginal due to the fact that the selected feature is likely, at least partially, redundant/correlated with the output of the trainable filters. Additional means of augmenting with more global data, such as the average grain size or equivalently the grain density, via inputs concatenated to the pooling layer (gray in Fig. 7) output going to the dense layers (green in Fig. 7) would also likely prove beneficial.

4.7 Generalizability

Using CNN, GCNN+, and GCNN# with 32 filters, one or two convolutional layers, and one dense layer, Fig. 16 shows that the proposed architecture with the # filter with an independent central weight can outperform a corresponding CNN and GCNN+. The benefits of the more complex (32:2:1) configuration over the other (32:1:1) appear to be most significant for the

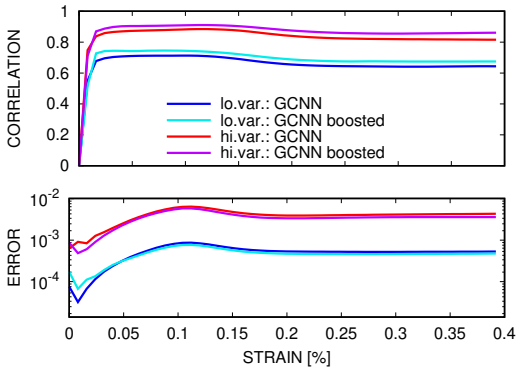
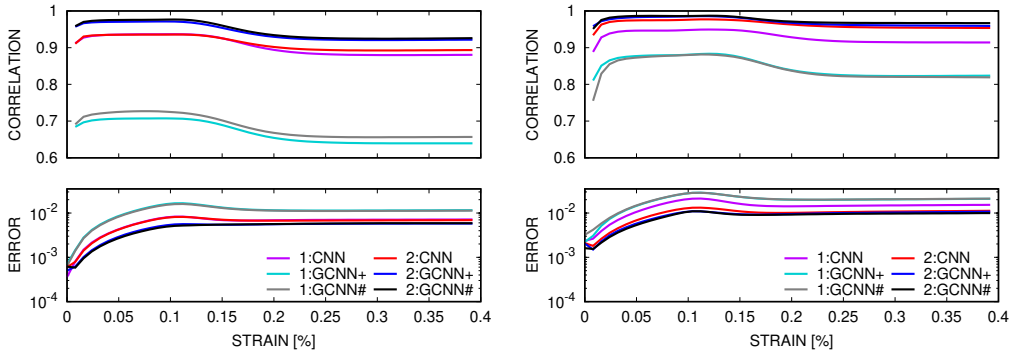


FIG. 15: Effect of boosting the microstructural input field with with volume fraction



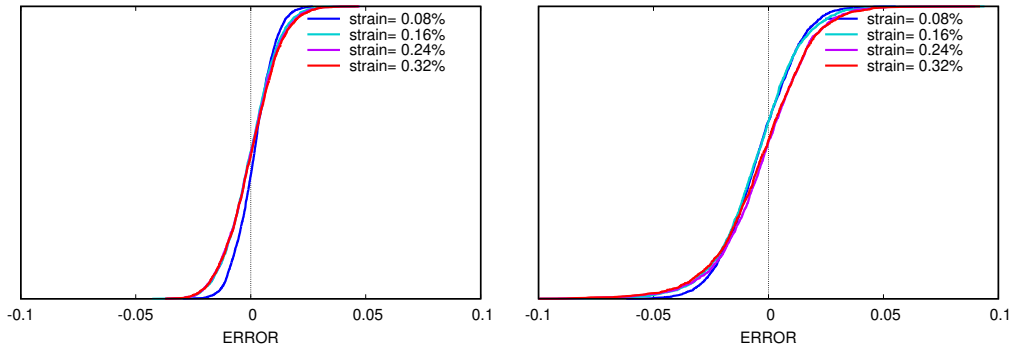
(a) (b)

FIG. 16: 3D data: performance of $(32{:}1{:}1)$ and $(32{:}2{:}1)$ configurations of a CNN, dGCNN+, and dGCNN# for the low- and high-variance datasets (2:GCNN+ denotes a $(32{:}2{:}1)$ direct graph CNN using a "+" pattern). (a) Low-variance and (b) high-variance.

two dGCNNs. It is also apparent that all the types of convolutional neural networks perform better, at least in terms of correlation, on the higher variance dataset than on the lower variance dataset.

Now focusing on the GCNN#, Fig. 17 shows the distribution of RMSE errors [Eq. (18)] is approximately Gaussian with some outlier errors above 5% for the high-variance dataset and 3% for the low-variance dataset. A direct comparison of the true and predicted values over a sequence of strains, shown in Fig. 18, indicates that the GCNN# overpredicts values near the mean, which may be due it being harder to distinguish near-mean response microstructures from those that produce extreme/outlier responses.

Following this conjecture, Fig. 19 illustrates that training on the low-variance ensemble and testing on the high-variance ensemble (which also has a different mean) does relatively poorly compared to the reverse. It appears that the network generalizes well to different distributions of inputs if they are in the span of the training set, i.e., it does well at interpolation and less well at extrapolation to potential out-of-distribution samples.



(a) (b)

FIG. 17: 3D data: cumulative distribution of error for various strains [$(32{:}2{:}1)$ dGCNN#]. (a) Low-variance and (b) high-variance.

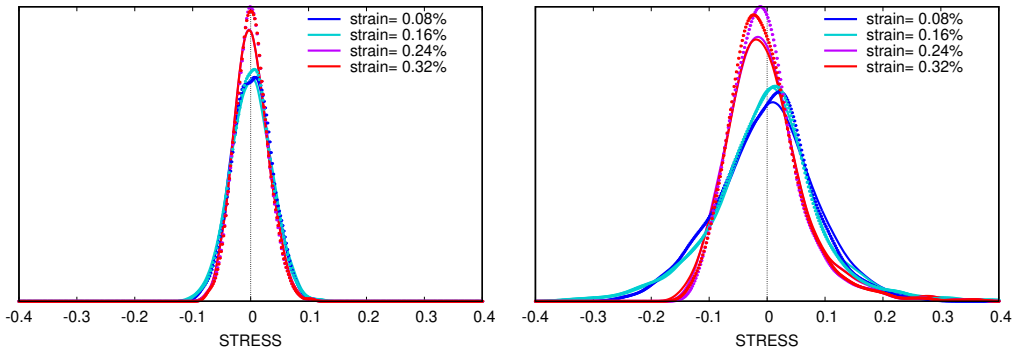


FIG. 18: 3D data: distribution of true (solid) and predicted (dotted) stress values for various strains [(32:2:1) dGCNN#]. (a) Low-variance and (b) high-variance.

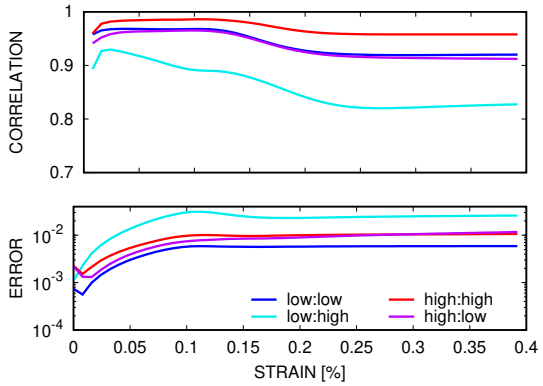


FIG. 19: 3D data: generalizability results for train:test pairs [(32:2:1) dGCNN#]

5. DISCUSSION

The response of microstructures for use in multiscale simulations, structure-property investigations, and uncertainty quantification can be accurately modeled with graphs. The proposed formulation used the topology of the data discretization directly instead of a segmentation or clustering of the image data. This aspect should have particular advantages for image data where the segmentation is not obvious, hard to compute, or obscured by noise. Furthermore, it has a simple implementation and avoids the need for feature engineering, but can benefit from it. The architecture draws on both purely graph-based networks and permutationally invariant convolutional filters. We demonstrated that endowing the widely used GCN filter (Kipf and Welling, 2016a) with an independent self-weight, as suggested by the reduction of the ChebNet, can significantly improve accuracy without adding additional layers and their parameters. The independent self-weight allows for differencing the node data of the self and its neighbors instead of only averaging. This can be seen as giving the filter the ability to infer edge features between the central pixel and its neighbors. For physical problems driven by gradients this change to the filter is important. We also found that pixel edge neighbors are more crucial for a predictive model than vertex-only neighbors. Lastly, we were able to demonstrate that small, efficient

graph convolutional networks can be effective at the task of predicting the homogenized evolution of complex microstructure. This has significant applications in subgrid constitutive models in large scale-simulations, structure-property investigations, and material uncertainty quantification.

An apparent downside of the proposed approach is the graph and its adjacency grows with resolution of image (number of pixels/elements). This issue is partially offset by sparse storage of the adjacency matrix, in general, and largely ameliorated by data that is on the same discretization. In future work we will investigate low-rank approximations to the adjacency matrix (Kanada et al., 2018; Lebedev et al., 2014; Richard et al., 2012; Savas and Dhillon, 2011; Tai et al., 2015), dimensionality reduction techniques (Belkin and Niyogi, 2003; He and Niyogi, 2004), and the use of graph autoencoders (Hasanzadeh et al., 2019; Kipf and Welling, 2016b; Liao et al., 2016; Salehi and Davulcu, 2019) to reduce the mesh-based graphs in-line. We are also pursuing the larger topic of processing images with multiresolution filters (Zhang et al., 2018), e.g., spanning the pixel to the cluster level.

ACKNOWLEDGMENTS

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. The authors gratefully acknowledge the use of the TensorFlow (https://www.tensorflow.org/) and Spektral (https://graphneural.network/) frameworks in this work.

REFERENCES

Albawi, S., Mohammed, T.A., and Al-Zawi, S., Understanding of a Convolutional Neural Network, in *Proc. of 2017 Int. Conf. on Engineering and Technology (ICET)*, Antalya, Turkey, August 21–23, 2017.

Belkin, M. and Niyogi, P., Laplacian Eigenmaps for Dimensionality Reduction and Data Representation, *Neural Comput.*, vol. **15**, no. 6, pp. 1373–1396, 2003.

Bessa, M.A., Bostanabad, R., Liu, Z., Hu, A., Apley, D.W., Brinson, C., Chen, W., and Liu, W.K., A Framework for Data-Driven Analysis of Materials under Uncertainty: Countering the Curse of Dimensionality, *Comput. Methods Appl. Mech. Eng.*, vol. **320**, pp. 633–667, 2017.

Bishop, C.M., *Pattern Recognition and Machine Learning*, Berlin/Heidelberg, Germany: Springer, 2006.

Bishop, J. and Hill, R., CXXVIII. A Theoretical Derivation of the Plastic Properties of a Polycrystalline Face-Centred Metal, *London, Edinburgh, Dublin Philosoph. Mag. J. Sci.*, vol. **42**, no. 334, pp. 1298–1307, 1951a.

Bishop, J. and Hill, R., XLVI. A Theory of the Plastic Distortion of a Polycrystalline Aggregate under Combined Stresses, *London, Edinburgh, Dublin Philosoph. Mag. J. Sci.*, vol. **42**, no. 327, pp. 414–427, 1951b.

Bjorck, J., Gomes, C., Selman, B., and Weinberger, K.Q., Understanding Batch Normalization, 2018. arXiv: 1806.02375

Bouquerel, J., Verbeken, K., and De Cooman, B., Microstructure-Based Model for the Static Mechanical Behavior of Multiphase Steels, *Acta Mater.*, vol. **54**, no. 6, pp. 1443–1456, 2006.

Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P., Geometric Deep Learning: Going beyond Euclidean Data, *IEEE Signal Proc. Mag.*, vol. **34**, no. 4, pp. 18–42, 2017.

Brown, J., Carroll, J., Huddleston, B., Casias, Z., and Long, K., A Multiscale Study of Damage in Elastomeric Syntactic Foams, *J. Mater. Sci.*, vol. **53**, no. 14, pp. 10479–10498, 2018.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y., Spectral Networks and Locally Connected Networks on Graphs, 2013. arXiv: 1312.6203

Čekada, M., Panjan, P., Kek-Merl, D., Panjan, M., and Kapun, G., SEM Study of Defects in PVD Hard Coatings, *Vacuum*, vol. **82**, no. 2, pp. 252–256, 2007.

Chen, G., Hong, Y., Zhang, Y., Kim, J., Huynh, K.M., Ma, J., Lin, W., Shen, D., Yap, P.T., and the UNC/UMN Baby Connectome Project Consortium, Estimating Tissue Microstructure with Undersampled Diffusion Data via Graph Convolutional Neural Networks, in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Lima, Peru, pp. 280–290, 2020.

Chidester, B., Do, M.N., and Ma, J., Rotation Equivariance and Invariance in Convolutional Neural Networks, 2018. arXiv: 1805.12301

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y., Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation, 2014. arXiv: 1406.1078

Chowdhury, A., Kautz, E., Yener, B., and Lewis, D., Image Driven Machine Learning Methods for Microstructure Recognition, *Comput. Mater. Sci.*, vol. **123**, pp. 176–187, 2016.

Cohen, T. and Welling, M., Group Equivariant Convolutional Networks, in *Proc. of Int. Conf. on Machine Learning*, pp. 2990–2999, New York, NY, June 19–24, 2016.

Dawson, P.R., Computational Crystal Plasticity, *Int. J. Solids Struct.*, vol. **37**, nos. 1-2, pp. 115–130, 2000.

DeCost, B.L., Francis, T., and Holm, E.A., Exploring the Microstructure Manifold: Image Texture Representations Applied to Ultrahigh Carbon Steel Microstructures, *Acta Mater.*, vol. **133**, pp. 30–40, 2017.

Defferrard, M., Bresson, X., and Vandergheynst, P., Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering, 2016. arXiv: 1606.09375

Dieleman, S., De Fauw, J., and Kavukcuoglu, K., Exploiting Cyclic Symmetry in Convolutional Neural Networks, in *Proc. of 33rd Int. Conf. on Machine Learning*, New York, NY, pp. 1889–1898, June 19–24, 2016.

Finzi, M., Stanton, S., Izmailov, P., and Wilson, A.G., Generalizing Convolutional Neural Networks for Equivariance to Lie Groups on Arbitrary Continuous Data, in *Proc. of Int. Conf. on Machine Learning*, Virtual Event, pp. 3165–3176, July 13–18, 2020.

Frankel, A.L., Jones, R.E., Alleman, C., and Templeton, J.A., Predicting the Mechanical Response of Oligocrystals with Deep Learning, *Comput. Mater. Sci.*, vol. **169**, p. 109099, 2019.

Frankel, A., Tachida, K., and Jones, R., Prediction of the Evolution of the Stress Field of Polycrystals Undergoing Elastic-Plastic Deformation with a Hybrid Neural Network Model, *Mach. Learn.: Sci. Technol.*, vol. **1**, no. 3, p. 035005, 2020.

Fulton, L., Modi, V., Duvenaud, D., Levin, D.I., and Jacobson, A., Latent-Space Dynamics for Reduced Deformable Simulation, *Computer Graphics Forum*, vol. **38**, Hoboken, NJ: Wiley Online Library, pp. 379–391, 2019.

Ghosh, S. and Dimiduk, D., Eds., *Computational Methods for Microstructure-Property Relationships*, vol. **1**, Berlin/Heidelberg, Germany: Springer, 2011.

Glorot, X. and Bengio, Y., Understanding the Difficulty of Training Deep Feedforward Neural Networks, in *Proc. of the 13th Int. Conf. on Artificial Intelligence and Statistics*, pp. 249–256, Sardinia, Italy, May

13–15, 2010.

Glorot, X., Bordes, A., and Bengio, Y., Deep Sparse Rectifier Neural Networks, in *Proc. of the 14th Int. Conf. on Artificial Intelligence and Statistics*, pp. 315–323, Ft. Lauderdale, FL, USA, April 11–13, 2011.

Goodfellow, I., Bengio, Y., and Courville, A., *Deep Learning*, Cambridge, MA: MIT Press, 2016.

Groeber, M.A. and Jackson, M.A., *DREAM.3D*, accessed September 2020, from http://dream3d.bluequartz.net, 2019.

Hammond, D.K., Vandergheynst, P., and Gribonval, R., Wavelets on Graphs via Spectral Graph Theory, *Appl. Comput. Harmonic Anal.*, vol. **30**, no. 2, pp. 129–150, 2011.

Hasanzadeh, A., Hajiramezanali, E., Duffield, N., Narayanan, K.R., Zhou, M., and Qian, X., Semi-Implicit Graph Variational Auto-Encoders, 2019. arXiv: 1908.07078

Hastie, T., Tibshirani, R., Friedman, J., and Franklin, J., The Elements of Statistical Learning: Data Mining, Inference and Prediction, *Math. Intel.*, vol. **27**, no. 2, pp. 83–85, 2005.

He, X. and Niyogi, P., Locality Preserving Projections, *Advances in Neural Information Processing Systems 16*, pp. 153–160, Vancouver, Canada, December 8–13, 2004.

Heckman, N.M., Ivanoff, T.A., Roach, A.M., Jared, B.H., Tung, D.J., Brown-Shaklee, H.J., Huber, T., Saiz, D.J., Koepke, J.R., Rodelas, J.M., Madison, J.D., Salzbrenner, B.C., Swiler, L.P., Jones, R.E., and Boyce, B.L., Automated High-Throughput Tensile Testing Reveals Stochastic Process Parameter Sensitivity, *Mater. Sci. Eng.: A*, vol. **772**, p. 138632, 2020.

Herriott, C. and Spear, A.D., Predicting Microstructure-Dependent Mechanical Properties in Additively Manufactured Metals with Machine- and Deep-Learning Methods, *Comput. Mater. Sci.*, vol. **175**, p. 109599, 2020.

Hochreiter, S. and Schmidhuber, J., Long Short-Term Memory, *Neural Comput.*, vol. **9**, no. 8, pp. 1735–1780, 1997.

Hopfield, J.J., Neural Networks and Physical Systems with Emergent Collective Computational Abilities, *Proc. National Acad. Sci.*, vol. **79**, no. 8, pp. 2554–2558, 1982.

Jones, R., Templeton, J.A., Sanders, C.M., and Ostien, J.T., Machine Learning Models of Plastic Flow Based on Representation Theory, *Comput. Model. Eng. Sci.*, vol. **117**, no. 3, pp. 309–342, 2018.

Kanada, T., Onuki, M., and Tanaka, Y., Low-Rank Sparse Decomposition of Graph Adjacency Matrices for Extracting Clean Clusters, in *Proc. of 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 1153–1159, Honolulu, Hawaii, November 12–15, 2018.

Khalil, M., Teichert, G., Alleman, C., Heckman, N., Jones, R., Garikipati, K., and Boyce, B., Modeling Strength and Failure Variability Due to Porosity in Additively Manufactured Metals, *Comput. Methods Appl. Mech. Eng.*, vol. **373**, p. 113471, 2021.

Kipf, T.N. and Welling, M., Semi-Supervised Classification with Graph Convolutional Networks, 2016a. arXiv: 1609.02907

Kipf, T.N. and Welling, M., Variational Graph Auto-Encoders, 2016b. arXiv: 1611.07308

Kocks, U., Laws for Work-Hardening and Low-Temperature Creep, *J. Eng. Mater. Technol.*, vol. **98**, no. 1, pp. 76–85, 1976.

Kondo, R., Yamakawa, S., Masuoka, Y., Tajima, S., and Asahi, R., Microstructure Recognition Using Convolutional Neural Networks for Prediction of Ionic Conductivity in Ceramics, *Acta Mater.*, vol. **141**, pp. 29–38, 2017.

Kondor, R. and Trivedi, S., On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups, in *Proc. of 35th Int. Conf. on Machine Learning*, Stockholm, Sweden, pp. 2747–2755, July 10–15, 2018.

Kraft, T., Rettenmayr, M., and Exner, H., An Extended Numerical Procedure for Predicting Microstructure

and Microsegregation of Multicomponent Alloys, *Model. Simul. Mater. Sci. Eng.*, vol. **4**, no. 2, p. 161, 1996.

Krizhevsky, A., Sutskever, I., and Hinton, G.E., Imagenet Classification with Deep Convolutional Neural Networks, *Adv. Neural Inform. Proc. Sys.*, vol. **25**, pp. 1097–1105, 2012.

Kroner, E., On the Plastic Deformation of Polycrystals, *Acta Metallurg.*, vol. **9**, no. 2, pp. 155–161, 1961.

Le, C., Bruns, T.E., and Tortorelli, D.A., Material Microstructure Optimization for Linear Elastodynamic Energy Wave Management, *J. Mech. Phys. Solids*, vol. **60**, no. 2, pp. 351–378, 2012.

Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., and Lempitsky, V., Speeding-Up Convolutional Neural Networks Using Fine-Tuned CP-Decomposition, 2014. arXiv: 1412.6553

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., and Jackel, L.D., Backpropagation Applied to Handwritten Zip Code Recognition, *Neural Comput.*, vol. **1**, no. 4, pp. 541–551, 1989.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P., Gradient-Based Learning Applied to Document Recognition, *Proc. IEEE*, vol. **86**, no. 11, pp. 2278–2324, 1998.

Lee, K. and Carlberg, K., Deep Conservation: A Latent-Dynamics Model for Exact Satisfaction of Physical Conservation Laws, 2019. arXiv: 1909.09754

Li, Y., Hu, S., Sun, X., and Stan, M., A Review: Applications of the Phase Field Method in Predicting Microstructure and Property Evolution of Irradiated Nuclear Materials, *npj Comput. Mater.*, vol. **3**, no. 1, p. 16, 2017.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A., Multipole Graph Neural Operator for Parametric Partial Differential Equations, 2020. arXiv: 2006.09535

Liao, Y., Wang, Y., and Liu, Y., Graph Regularized Auto-Encoders for Image Representation, *IEEE Transact. Image Process.*, vol. **26**, no. 6, pp. 2839–2852, 2016.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., and Alsaadi, F.E., A Survey of Deep Neural Network Architectures and Their Applications, *Neurocomput.*, vol. **234**, pp. 11–26, 2017.

Lubbers, N., Lookman, T., and Barros, K., Inferring Low-Dimensional Microstructure Representations Using Convolutional Neural Networks, *Phys. Rev. E*, vol. **96**, no. 5, p. 052111, 2017.

Lubliner, J., *Plasticity Theory*, North Chelmsford, MA: Courier Corporation, 2008.

Mandel, J., Généralisation de la Théorie de Plasticité de WT Koiter, *Int. J. Solids Struct.*, vol. **1**, no. 3, pp. 273–295, 1965.

Mecking, H., Kocks, U., and Fischer, H., Hardening, Recovery, and Creep in FCC Mono- and Polycrystals, in *Proc. the 4th Int. Conf. on Strength of Metals and Alloys*, Nancy, France, August 30–September 3, 1976.

Mozaffar, M., Bostanabad, R., Chen, W., Ehmann, K., Cao, J., and Bessa, M., Deep Learning Predicts Path-Dependent Plasticity, *Proc. Nat. Acad. Sci.*, vol. **116**, no. 52, pp. 26414–26420, 2019.

Mura, T., *Micromechanics of Defects in Solids*, Berlin/Heidelberg, Germany: Springer Science & Business Media, 2013.

Nemat-Nasser, S. and Hori, M., *Micromechanics: Overall Properties of Heterogeneous Materials*, Amsterdam, Netherlands: Elsevier, 2013.

Noh, J., Kim, J., Stein, H.S., Sanchez-Lengeling, B., Gregoire, J.M., Aspuru-Guzik, A., and Jung, Y., Inverse Design of Solid-State Materials via a Continuous Representation, *Matter*, vol. **1**, no. 5, pp. 1370–1384, 2019.

O'Shea, K. and Nash, R., An Introduction to Convolutional Neural Networks, 2015. arXiv: 1511.08458

Pandey, A. and Pokharel, R., Machine Learning Enabled Surrogate Crystal Plasticity Model for Spatially Resolved 3D Orientation Evolution under Uniaxial Tension, 2020. arXiv: 2005.00951

Qin, Z., Yu, F., Liu, C., and Chen, X., How Convolutional Neural Network See the World—A Survey of Convolutional Neural Network Visualization Methods, 2018. arXiv: 1804.11191

Quiroga, F., Ronchetti, F., Lanzarini, L., and Bariviera, A.F., Revisiting Data Augmentation for Rotational Invariance in Convolutional Neural Networks, in *Modelling and Simulation in Management Sciences*, Berlin/Heidelberg, Germany: Springer, pp. 127–141, 2018.

Richard, E., Savalle, P.A., and Vayatis, N., Estimation of Simultaneously Sparse and Low Rank Matrices, 2012. arXiv: 1206.6474

Rizzi, F., Khalil, M., Jones, R.E., Templeton, J.A., Ostien, J.T., and Boyce, B.L., Bayesian Modeling of Inconsistent Plastic Response Due to Material Variability, 2018. arXiv: 1809.01009

Roduit, C., Sekatski, S., Dietler, G., Catsicas, S., Lafont, F., and Kasas, S., Stiffness Tomography by Atomic Force Microscopy, *Biophys. J.*, vol. **97**, no. 2, pp. 674–677, 2009.

Roters, F., Eisenlohr, P., Hantcherli, L., Tjahjanto, D.D., Bieler, T.R., and Raabe, D., Overview of Constitutive Laws, Kinematics, Homogenization and Multiscale Methods in Crystal Plasticity Finite-Element Modeling: Theory, Experiments, Applications, *Acta Mater.*, vol. **58**, no. 4, pp. 1152–1211, 2010.

Rowley, H.A., Baluja, S., and Kanade, T., Rotation Invariant Neural Network-Based Face Detection, in *Proc. of 1998 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, pp. 38–44, Santa Barnara, California, June 23–25, 1998.

Salehi, A. and Davulcu, H., Graph Attention Auto-Encoders, 2019. arXiv: 1905.10715

Salinger, A.G., Bartlett, R.A., Bradley, A.M., Chen, Q., Demeshko, I.P., Gao, X., Hansen, G.A., Mota, A., Muller, R.P., Nielsen, E., Ostien, J.T., Pawlowski, R.P., Perego, M., Phipps, E.T., Sun, W.C., and Tezaur, I.K., Albany: Using Component-Based Designto Develop a Flexible, Generic Multiphysics Analysis Code, *Int. J. Multiscale Comput. Eng.*, vol. **14**, no. 4, pp. 415–438, 2016.

Savas, B. and Dhillon, I.S., Clustered Low Rank Approximation of Graphs in Information Science Applications, in *Proc. of the 2011 SIAM International Conference on Data Mining*, Mesa, AZ, USA, pp. 164–175, April 28–30, 2011.

Shi, X., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K., and Woo, W., Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting, in *Proc. of Advances in Neural Information Processing Systems 28: Annual Conf. on Neural Information Processing Systems*, Montreal, Canada, pp. 802–810, December 7–12, 2015.

Silhavy, M., *The Mechanics and Thermodynamics of Continuous Media*, Berlin/Heidelberg, Germany: Springer Science & Business Media, 2013.

Stenzel, O., Pecho, O., Holzer, L., Neumann, M., and Schmidt, V., Predicting Effective Conductivities Based on Geometric Microstructure Characteristics, *AIChE J.*, vol. **62**, no. 5, pp. 1834–1843, 2016.

Stewart, J.R. and Edwards, H.C., Sierra Mechanics, accessed September 2020, from https://www.sandia.gov/asc/advanced-simulation-and-computing/integrated-codes, 2020.

Tai, C., Xiao, T., Zhang, Y., Wang, X., and E, W., Convolutional Neural Networks with Low-Rank Regularization, 2015. arXiv: 1511.06067

Taylor, G.I., The Mechanism of Plastic Deformation of Crystals. Part I. Theoretical, *Proc. Roy. Soc. London. Ser. A*, vol. **145**, no. 855, pp. 362–387, 1934.

Trask, N., Huang, A., and Hu, X., Enforcing Exact Physics in Scientific Machine Learning: A Data-Driven Exterior Calculus on Graphs, 2020. arXiv: 2012.11799

Trask, N., Patel, R.G., Gross, B.J., and Atzberger, P.J., GMLS-Nets: A Framework for Learning from Unstructured Data, 2019. arXiv: 1909.05371

Trovalusci, P., Capecchi, D., and Ruta, G., Genesis of the Multiscale Approach for Materials with Microstructure, *Archive Appl. Mech.*, vol. **79**, no. 11, pp. 981–997, 2009.

Vlassis, N.N., Ma, R., and Sun, W., Geometric Deep Learning for Computational Mechanics Part I:

Anisotropic Hyperelasticity, *Comput. Methods Appl. Mech. Eng.*, vol. **371**, p. 113299, 2020.

Worrall, D.E., Garbin, S.J., Turmukhambetov, D., and Brostow, G.J., Harmonic Networks: Deep Translation and Rotation Equivariance, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, pp. 5028–5037, July 21–26, 2017.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S.Y., A Comprehensive Survey on Graph Neural Networks, *Proc. IEEE Transact. Neural Networks Learn. Syst.*, vol. **32**, no. 1, pp. 4–24, 2020.

Yin, X., Chen, W., To, A., McVeigh, C., and Liu, W.K., Statistical Volume Element Method for Predicting Microstructure—Constitutive Property Relations, *Comput. Methods Appl. Mech. Eng.*, vol. **197**, nos. 43-44, pp. 3516–3529, 2008.

Zhang, T., Liu, B., Niu, D., Lai, K., and Xu, Y., Multiresolution Graph Attention Networks for Relevance Matching, *Proc. of the 27th ACM Int. Conf. on Information and Knowledge Management*, Torino, Italy, pp. 933–942, October 22–26, 2018.

Zhang, Z., Cui, P., and Zhu, W., Deep Learning on Graphs: A Survey, 2020. arXiv: 1812.04202

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M., Graph Neural Networks: A Review of Methods and Applications, *AI Open*, vol. **1**, pp. 57–81, 2020.

APPENDIX A. THE GRAPH CONVOLUTIONAL NETWORK

As mentioned in the Introduction, the Kipf and Welling graph convolutional network (GCN) (Kipf and Welling, 2016a) provides an innovative, expressive graph convolutional network built in sequentially applied layers with local action. Here we give a brief synopsis of their development.

The objective is to efficiently apply a graph filter $g_\theta = \mathrm{diag}(\theta)$ where the parameter vector θ has N_{nodes} entries. Convolution of the filter g_θ and data x on a graph can be expressed as (Hammond et al., 2011)

$$g_\theta * x = U g_\theta U^T x, \tag{A.1}$$

analogous to the classical convolution theorem. This formulation is connected to the (normalized) graph Laplacian L and its spectral representation

$$L = I - D^{-1/2} A D^{-1/2} = U \Lambda U^T, \tag{A.2}$$

where A is the binary adjacency matrix, D is the associated degree matrix, U is the matrix of eigenvectors, and Λ is the diagonal matrix of eigenvalues. The formulation for graph convolution in Eq. (A.1), in turn, can be approximated by an expansion of Chebyshev polynomials T_k (Defferrard et al., 2016)

$$g_\theta * x = \left(U g_\theta U^T\right) x \approx \sum_k \vartheta_{k=0}^K T_k(\tilde{L})\, x, \tag{A.3}$$

where $\tilde{L} = (2/\lambda_{\text{max}})L - I$ and λ_{max} is the maximum eigenvalue of L.

To this approximation Kipf and Welling (2016a) make a number of additional simplifications. First they approximate the maximum eigenvalue $\lambda_{\text{max}} \approx 2$ so that $\tilde{L} = L - I$, i.e., $\tilde{L}$ is the graph Laplacian with added self-loops/interactions. Next, they truncate the expansion in Eq. (A.3) at $K = 1$ so that

$$g_\theta * x \approx \vartheta_0 I x - \vartheta_1 D^{-1/2} A D^{-1/2} x. \tag{A.4}$$

This effectively reduces the number of free parameters in the filter from N_{nodes} to 2. This has the tremendous advantage of cheap and local action. The expressiveness of a network built on these layers is controllable by the GCNN depth (number of layers). Lastly, they further collapse the number of free parameters from 2 to 1 by setting $\vartheta_1 = -\vartheta_0$.