A Convolutional Neural Network for Enhancement of Multi-Scale Localization in Granular Metallic Representative Unit Cells

Peter A. Gustafson[1,2]
Western Michigan University, Kalamazoo, MI, 49008, USA

NASA Glenn Research Center, Cleveland, OH, 44135, USA

Evan J. Pineda[3] and Trenton M. Ricks[4] and Brett A. Bednarcyk[5] and Brandon L. Hearley[6] and Joshua Stuckner[7]
NASA Glenn Research Center, Cleveland, OH, 44135, USA

A convolutional neural network was used to enhance the localization of strain and stress for a generalized method of cells model of a metallic microstructure. Enhanced shear strains, measured in terms of the linear regression coefficients as a function of ground truth strains, were improved from inaccurate and uncorrelated (slope=0.003, r^2=0.000) to accurate and well correlated (slope=0.890, r^2=0.882) relative to ground truth (slope=1.0, r^2=1.0). In applying the convolutional neural network, a convolutional stride of 1.0 (padding='same') was only modestly effective while strides of 2 or 3 were more effective yet at higher cost. Additional convolutional layers were generally more expensive than additional dense layers, often with limited benefit. The accuracy of enhanced localized shear strains and stress is expected to yield benefits for damage progression models, especially in the context of hierarchical multi-scale methods where the generalized method of cells is applied at the intermediate scale.

I. Nomenclature

ANN	=	artificial neural network
CT	=	computed tomography
CNN	=	convolutional neural network
FE	=	finite element
GMC	=	generalized method of cells
HFGMC	=	high fidelity generalized method of cells
ICME	=	integrated computational materials engineering
MAC/GMC	=	Micromechanics Analysis Code with Generalized Method of Cells

[1] Professor of Mechanical and Aerospace Engineering, AIAA Senior Member
[2] Visiting Scholar, Materials and Structures Division
[3] Research Aerospace Engineer, Materials and Structures Division, AIAA Senior Member
[4] Research Aerospace Engineer, Materials and Structures Division, AIAA Member
[5] Research Materials Engineer, Materials and Structures Division, AIAA Associate Fellow
[6] Structures/Materials Research Scientist, Materials and Structures Division
[7] Materials Informatics Scientist, Materials and Structures Division

ML = machine learning
MRI = magnetic resonance imaging
NASMAT = NASA Multi-scale Analysis Tool
RUC = representative unit cell

II. Introduction

Multi-scale modeling simulates the response of a system over multiple length or time scales. Machine learning (ML) is a subset of artificial intelligence in which algorithms are trained on large datasets to perform tasks without being explicitly programmed to do so.

Initial applications of machine learning that have already had significant commercial and research impact include image recognition and natural language processing. The breadth of potential applications of this technology is only beginning to be explored. Mechanical and aerospace engineering opportunities are numerous and are on the leading edge of the discipline. Some applications with the largest current impact include automation of vehicles (i.e., autopilot and automated driving), and biomedical diagnosis (classification based on CT and MRI imaging). However, the breadth of applications of machine learning is sure to increase in the coming years.

Multi-scale structural modeling is critical to enabling the next generation of Integrated Computational Materials Engineering (ICME), which links materials manufacturing and design in an integrated framework [1]–[4]. The challenge of these systems is the *heterogeneity and complexity* of the constituent behavior at all length scales and the interaction between the scales. Multi-scale structural analyses may be performed by letting a micro mechanics model act as the constitutive model at the integration points of a larger finite element model. This technique is considered accurate for homogenized elastic stiffness and can be accurate also for many damage micromechanics models. However, these come at a large computational cost especially in the context of in-elasticity or damage. Further, known limitations exist especially with respect to calculating localized normal-shear coupling via shear lag.

In the remaining subsections of this introduction, a limited description is provided of a machine learning technique that is used in this article (i.e., a convolutional neural network). Detailed technical descriptions of machine learning are left to other resources [5], [6]. Subsequently, some of the limitations of a multi-scale structural analysis technique (the generalized method of cells) are discussed in detail in order to motivate the work described thereafter. Finally, the goal of this article will be stated prior to descriptions of the methods and results.

Artificial neural networks and convolutional neural networks

A popular machine learning technique is the artificial neural network (ANN), which is made up of an input layer, any number of hidden layers, and an output layer. Each hidden layer consists of neurons which act to receive input from the previous layer, and generate an output based on the weighted sum of the inputs. The weights represent the strength of connections between neurons. Nonlinearity is introduced into the neural network by selecting the appropriate *activation functions*, which adjusts the output of each node based on a nonlinear function. Further, higher-order nonlinearity comes from applying multiple layers (with nonlinear activation functions) within the network. *Learning*, in this context, is the process of updating these weights based on an input data set, reference dataset, and training loss. In many cases, machine learning models can act as a 'black box' which accurately predicts output for a given input of a complex system but with minimal cost.

A novel approach was pioneered in [7] to treat composite fiber microstructure similarly to an image recognition and classification problem. A specialized algorithm called a convolutional neural network (CNN) was used which differs from a traditional ANN in that it accepts images as inputs. Through a series of convolutional and pooling layers, CNNs extract the relevant features from the image for downstream use. Convolutional filters in a CNN are analogous to weights in the connections of an ANN; they are updated throughout training. CNNs are traditionally used for classification, but can be modified to perform regression. Sorini et al. [7] demonstrated that a CNN can take images of composite microstructures as input and provide accurate predictions of linear elastic homogenized material response. Further, they demonstrated that the method is computationally fast relative to physics based simulation alternatives.

Expanding on the method described in [7], this article treats a micro-scale generalized method of cells (GMC) in a manner similar to a *3D image*, where each subcell is treated like a voxel in a 3D convolutional neural network. Finite element models are used as ground truth to train the network such that the network accepts GMC input and provides “improved” or “corrected” localized stresses and strains within the subcells of the model. The next section introduces GMC and the reasons that corrections to its stress prediction are sought.

Limitations of the generalized method of cells

A multi-scale analysis technique that has proven effective for elastic and some damage mechanics modeling is the generalized method of cells (GMC) [8]. GMC analyzes representative unit cells (RUC) which are composed of an appropriate number of subcells that represent features of the microstruture. This has been used in multi-scale analysis of 2D fibrous composite analysis, interfacial damage, triaxially braided composites, inelastic and viscoelastic-viscoplastic analysis, single crystal plasticity, as well as other uses. [8]–[16]

Validation of the generalized method of cells (GMC) micromechanics theory [17] has been done with reference to the finite element method, but this validation has predominantly focused on the macroscopic stiffness outcome. Comparatively fewer validations have been done examining localization of all stresses and strains. Similarly, more extensive validation has been done in two dimensions than in three dimensions. However, the comparisons to the finite element method have been favorable under appropriate circumstances and considering intended outcomes. The primary limitation of the generalized method of cells comes from the assumption of *constant normal stress* in the RUC within its assumed coordinate frame. As a consequence and due to stress equilibrium, it is also assumed that there can be *no variation of shear stress in the model*. Thus it is common to state that *GMC lacks of normal-shear coupling via shear lag*. Furthermore, it is also assumed that the boundary conditions are periodic. Because of the assumptions, *typical analysis strives to provide a homogenized response that is macroscopically accurate in the average sense over the representative unit cell (RUC)*. Though these assumptions provide for a comparatively quick analysis tool, they nevertheless impose restrictions on the localization of stress and strain, and thus potentially also restrict the damage mechanics that would be competed at the local and sub-local scale [18].

In contrast to GMC, the *high fidelity generalized method of cells* (HFGMC) provides for a normal stress field that can vary spatially within the representative unit cell. This field varies assuming a second order subcell displacement field, which results in stresses and strains that vary linearly within the subcells. Normal-shear coupling (via shear lag) can thus be modeled within the RUC. Despite the coupled fields within the RUC, the method still *assumes* periodicity in the boundary conditions. This manifests as the tractions on each boundary subcell face being equal to corresponding tractions on the opposite boundary subcell face of a RUC. The assumed boundary periodicity can be demonstrated to be imprecise when there is internal asymmetry within the RUC, and thus imposes some degree of inaccuracy in the calculated RUC fields. While the absolute "error" that can manifest as a result of the assumptions varies depending on the symmetry of the RUC, and while the relative error this contributes compared to the overall discretization and modeling strategy (as with all models), the assumption of periodicity results in an internal RUC stress field that functionally violates the differential equations of elasticity to some degree. Nevertheless, both GMC and HFGMC *may be accurate enough for the purposes of the intended model and facilitate a relatively inexpensive localized solution compared to finite element methods*. In summary, HFGMC can model internal fields more accurately than GMC, but at greater computational cost. Consequently, HFGMC may be more able to capture damage mechanics accurately, especially if localized damage mechanics depend heavily on localized shear stress. The described assumptions, benefits, and costs apply to 2-dimensional and 3-dimensional models, as well as all material models when the method of cells [17] family of algorithms is used.

Method of cells limitations with respect to damage mechanics

Micro-mechanics models are considered especially valuable when micro-scale localized damage phenomena manifest and where homogenized damage criteria fail. Damage criteria typically use tensor invariants to predict onset and propagation of damage. Hence, localized invariants are of great significance in evaluating the predicted response. It has been shown that the stress and strain invariants typically used for homogenized damage models (such as von Mises equivalent stress) can be accurately captured by GMC and HFGMC. However, this finding cannot be separated from the assumption of constant normal stresses within the RUC (or traction symmetry in HFGMC). If the normal stresses are constant and the shears are zero or meaningless in GMC, the spatial distribution of invariants may not be accurately localized within GMC. Even if the invariants are accurately captured in the RUC-averaged sense without accurate computation of shear, *the implications of constant normal stresses in GMC on localized damaged mechanics and plasticity are not well understood*. Consider a hierarchical multi-scale analysis where GMC is part of the chain of scales. Assume the highest analysis length scale is resolved using finite elements, GMC acts as the effective constitutive law at the finite element integration points model, and any appropriate method predicts the response at a small scale that impacts the GMC constitutive prediction scale (GMC, HFGMC, FEA, etc). In this three scale model, the micro-scale (intermediate scale) GMC limitations constrain the analysis at

the smallest scale, as well as the selection of subcells (i.e., subcell filtering or downselection) that require analysis at the smallest scale.

For efficiency, it is likely that the models of the smallest scale must be downselected from subcells of the intermediate scales. Downselection requires criteria, and it is also likely that the criteria evaluate localized scalar invariants. For example, J2 metal plasticity does not consider microstructural features, and consequently provides a homogenized approach in the scale at which it is applied. Similarly, Tsai-Wu is a homogenized scalar composite failure criterion. Neither is ideally suited for application within a micromechanics model, as the purpose of a multi-scale model is to capture the underlying physics of localized damage. Nevertheless, these are natural candidates for a subcell downselection criteria as their purpose is to predict likely damage progression based on features not captured at the current feature scale. *The constant normal stresses and resulting imprecise distribution of stress invariants within GMC may limit the use of these well known criteria in down-selection, as well as limit the accuracy of current-scale damage modeling. Any methodology that can provide a correction to GMC without substantial additional cost may significantly improve its utility in damage prediction and down-selection filtering.* Though HFGMC provides for spatially non-constant invariants in the subcells and is thus an improvement over GMC, its increased cost may be prohibitive.

The proposed method of this paper intends to improve the stress and strain prediction of GMC with only a minor increment in cost. Consequently, localized damage mechanics can be more accurately evaluated in the local/current scale as well as across the hierarchical scales when down selection must be applied.

Goal of this article

The trade-offs between cost and fidelity should always be considered in analysis. Given the recent advances in ML for generating surrogate models for physical and other processes, there is an opportunity to seek compromise between cost and fidelity in the method of cells [17] family of algorithms. Alternatively, in some cases, trained and validated ML surrogate can replace the functionality of a higher fidelity model at low cost.

This article focuses on the GMC and its limitations, and intends to demonstrate that a ML surrogate can be used as a "correction" to the limitations of the GMC kinetic fields. More specifically, the lack of normal-shear coupling via shear lag may be circumvented with machine learning. This article proposes a methodology for improving the localization outcome at a small computational cost compared to more precise techniques.

III. Method

Training data: generation of finite element and GMC models

GMC was used to create 3D models of metallic crystal microstructures. NASMAT version 3.0 alpha (developers version) was used as the GMC analysis software. Corresponding higher fidelity models were created using the CalculiX finite element solver of the same microstructures.

The grain microstructures shown in Figure 1 were established using a modified version of MicroStructPy [19] as well as custom python scripts. The MicroStructPy code used random seed positions to create 3D Voronoi diagrams. These diagrams provided the grain boundaries assumed in the microstructural models. From these boundaries, multiple meshes of the same geometry at various mesh densities were exported. (See Figure 1). Although finite element mesh exports are a feature set of MicroStructPy, it was determined that code modifications were necessary to yield functional meshes. Trivially, the node numbers were updated to write as integers rather than floats. More significantly, second order tetrahedral meshes were implemented in place of the first order element sets implemented in the code base. The advantages of second order meshes (higher accuracy at lower mesh resolutions) are well understood, as are the costs (higher degree of freedom counts for the same number of elements and longer run times). To achieve the second order conversion, the linear mesh was exported from MicroStructPy using its gmsh interface (version 4.8.4) [20] and gmsh's own python API. Subsequently, CalculiX CGX [21] was used via a script to promote the elements to second order and export new meshes.

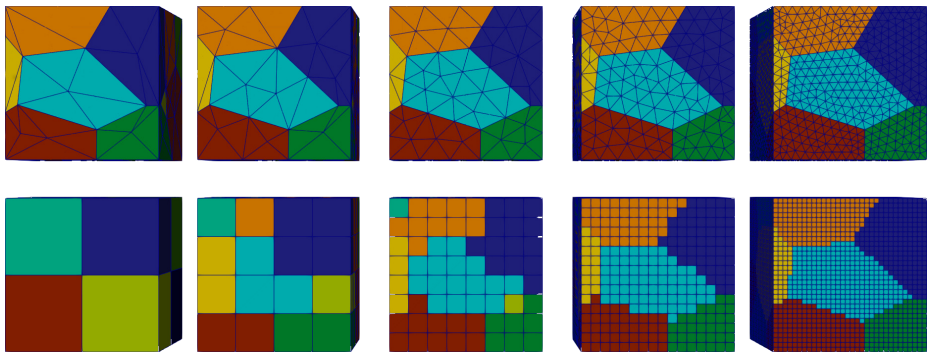


Figure 1: RUC microstructural models with varying numbers of subcells per model. The top row shows second order tetrahedral finite element meshes. The bottom row visualizes both the GMC microstructural representation and the voxelized finite element model (linear hexahedral C3D8 elements). The columns represent varying mesh densities of the same microstructure. ML models were trained for all of these mesh densities.

Custom Python scripts were subsequently used to identify facial boundary nodes on the second order meshes. Planar multi-point constraints were implemented, one for each of the six exterior faces of the cube. Eight total corner control nodes were used, three nodes per face, for the multi-point constraints. The eight corner control nodes were then used to apply 12 load steps, with each step representing macroscopic strain control of ±0.001 in one component of strain with all other components macroscopically 0. Hence, each of the six strain components were isolated and imposed in tension and compression in separate steps.

Consequently, after using MicroStructPy, CGX, and custom scripting, and by varying the random seed, 1000 unique microstructures were created as finite element models, and multiple mesh densities for each geometry were created. From these, additional python scripts were used to generate corresponding GMC input decks for NASMAT. GMC/NASMAT has hexahedral subcells (in 3D models, See Figure 2), hence, the GMC models were voxelized representations of the grain microstructure that follow the grain finite element boundary. The python scripts identified the voxel centers and determined in which microstructural grain the voxel center resided. From the GMC models, voxelized finite element models were created with hexahedral (C3D8) elements. Thus, the finalized finite element models and GMC models had corresponding subcells.

Cubic material properties (C11=168.4 GPa, C12=121.4 GPa, C44=75.4 GPa) were assumed at the grain level, and grain properties were specified such that the cubic principal directions of each grain were randomly oriented. The material tensors were transformed to align with the associated coordinate system and formatted separately for NASMAT or for CalculiX, and were entered as anisotropic material properties in the respective input deck formats. Hence, GMC and finite element models had identical material properties assigned to identical voxelizations. The principal difference between the model types was in the methodology (generalized method of cells vs finite element method), and thus there were moderate differences in the applied boundary conditions (FEA boundary conditions were planar multi point constraints with control nodes imposing macroscopic strain, GMC was subjected to imposed macroscopic strains where traction periodicity was also imposed automatically). All other aspects of the models were matched to the extent possible. The GMC models had the same functional 12 load steps applied in separate analysis, which were subsequently combined into a single output file in XDMF format, such that all 12 steps from a single output file and corresponding output files were available for finite element and GMC results.

In summary, for each of the GMC and FE methods described above, 1000 geometries were created with 12 linear elastic load steps each. These models provided a machine learning dataset of 12,000 model predictions. Further, to explore the effect of different levels of mesh density within the RUC, RUCs with $n \; x \; n \; x \; n$ subcells were created for each geometry, where n={2, 4, 8, 16, 32}. Hence, machine learning models were built with varying densities. FEA results from CalculiX were assumed to be ground truth and GMC were assumed as inputs to a model intending to reproduce ground truth from GMC models.

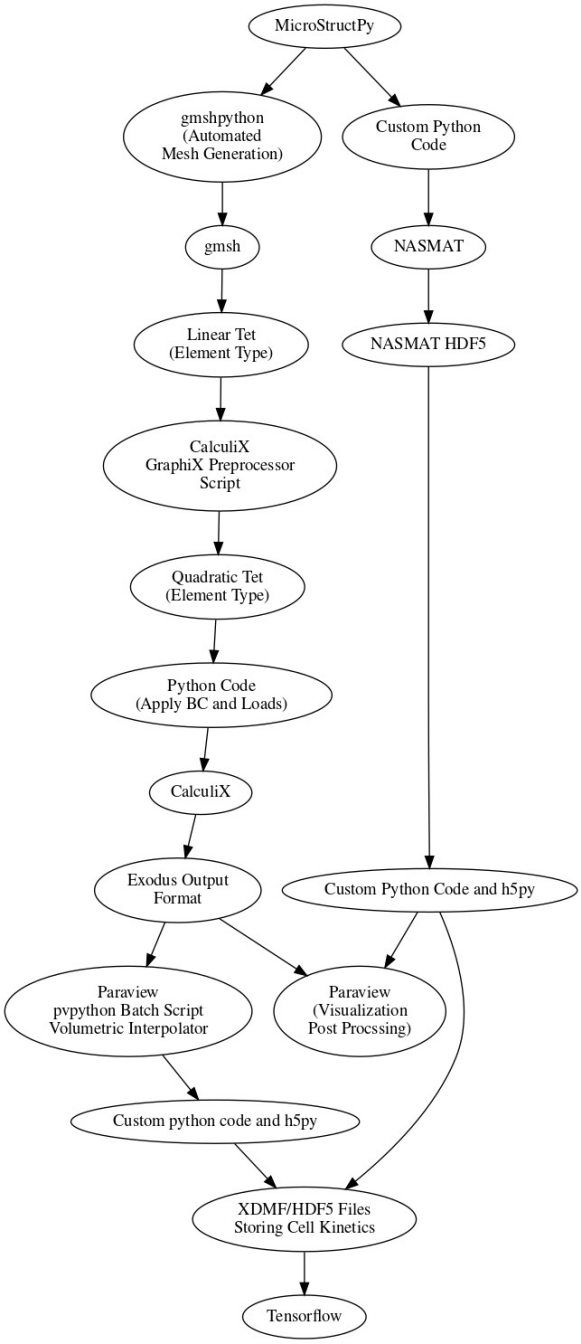


Figure 2: Diagramatic representation of the process for generating trained ML models

Training data: preprocessing

The training data was preprocessed for use with the Tensorflow machine learning platform. All preprocessing was done using standard Python scientific computing & machine learning libraries (including Numpy version 1.21.1 and Tensorflow version 2.5). Having been imposed at the ±0.001 level, strains were scaled by a factor of 1000 (thus, were approximately in the range of -1 to 1). Stresses were scaled by a factor of 1/1000 (and thus were also in the -1 to 1 order of magnitude). These scalings were completed to decrease the numerical disparity between stress and strain as consequently minimize the likelihood of non-convergence in the machine learning algorithms. The complete data sets were split into training (80%), validation (15%), and testing (5%) data categories.

Tensorflow's Keras module was used to produce the convolutional neural network architecture for machine learning (Figure 3 shows an example). The 'relu' activation function [22] was used for all neurons in each dense layer except the final dense layer. The final dense layer (i.e., the output layer) utilized a 'linear' activation function to permit a regression output including tensile and compressive strains and stresses. The loss function was 'mean squared error', and the optimizer was 'adam'. Many architectures were trained for comparison and evaluated with the validation data, each combining a number of 3D convolutional and dense layers. Both the input and the final output was a $n \; x \; n \; x \; n \; x \; 12$ tensor (where

n is the number of subcells in each direction and 12 reflects 6 strains and 6 stresses per cell) . The output tensors are evaluated with respect to the corresponding subcell ground truth FEA values using the loss function.

Having started with a general form of a CNN (i.e., one designed for regression and with identical input and output tensor dimensions), the network architecture was explored by varying the number of convolutional layers and dense layers. Convolutional layers were always applied first, varying the number of filters. Each convolutional layer was assigned the same number of filters. Different training sessions were assigned different numbers of layers and filters. To preserve the number of subcells, each convolutional layer was padded so its shape was held constant (padding='same'). Dense layers were subsequently applied where each dense layer (except the output layer) was assigned the same number of neurons. Again, different training sessions were assigned different numbers of neurons. To reduce the hyperparameter search space, the number of convolutional filters per layer and the number of neurons were always assigned to be an integer multiple of 12.

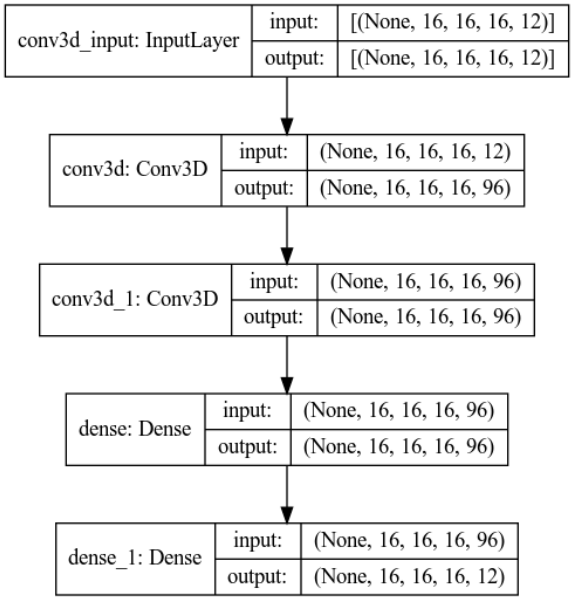


Figure 3: Example model architecture

Exploration was subsequently completed by varying the number of each type of layer and their parameters. An example architecture is shown in Figure 3. It has 3 convolutional layers with $n \; x \; 8 = 96$ filters, 1 dense layer with $n \; x \; 8 = 96$ neurons, and a final dense layer of 12 neurons for output compatibility. (Reported parameters are per subcell). This grid search approach to parameter optimization is inefficient, however, true optimization is the subject of future work.

Evaluation of the model was conducted in two ways. Training was based on minimization of the loss function. Hence, the training, testing, and validation losses are one measure of the fidelity of the trained network. However, as a more physically meaningful metric, the predicted strains and stresses were also plotted against the ground truth strains and stresses for each subcell of each RUC in the models. An ideal model would reproduce the ground truth, hence, a linear regression between the ground truth and model would yield a slope of 1 and an Pearson Correlation Coefficient (r^2) value of 1. Therefore, an ordinary least square linear slope and r^2 values were also used to quantitatively assess model quality. Scatter plots were generated for each ML model architecture and several are presented and discussed in the results section.

IV. Results

Finite element results

Figure 4 shows typical finite element model results for normal and shear strain and stress. In the figures, the applied load was a homogenized strain εxx=0.001. Due to the grain structure and local properties, the fields vary substantially within the volume. Recognizable patterns are visible, particularly in the 32x32x32 subcell RUCs on the right side of the figures. Note that GMC models possess only 1 localized subcell value (i.e., one strain value per component per subcell) whereas the finite element method has 8 integration point strains per component per element. (These are extrapolated to the nodes for visualization). For the purposes of the ML models, a paraview post-processing script was used to find interpolated values (i.e., averaged over the nodes of the subcells) for the finite element models. This provides a reasonable comparison between GMC and finite element models, and a foundation for the ML algorithms.

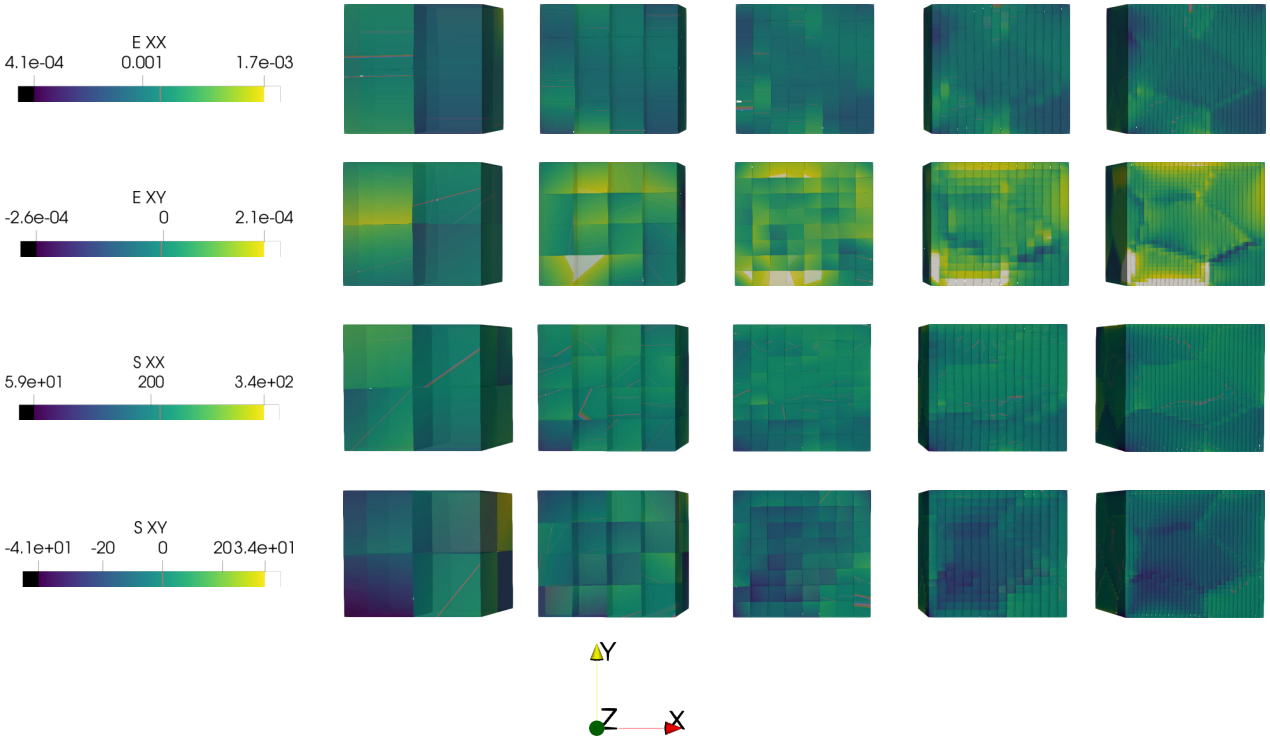


Figure 4. Finite element model results for strain and stress components at various RUC densities

The 1000 finite element models (x12 load steps) at each mesh density were used as ground truth. As seen in the figures, the loads will yield stresses and strains over a broad spectrum. The normal strains are predominantly close to the applied macroscopic strain (in the direction of the applied strain) or close to zero (since the homogenized transverse strain averages to zero). Excursions from |0.001| and 0 occur due to the cubic grain orientation redistributing load through the RUC. Similarly, due to the spatial variation in normal stress, shear stresses exist across a spectrum. For the applied shear load steps, similar arguments demonstrate an average of zero normal strain, but a non-zero average shear strain. Variations in shear stress lead to variations in normal stress. Hence, to assess the utility of the ML models presented below, they will be compared to the strains (and stresses) of the ground-truth FEA models.

In the remaining subsections, results will be described for the 8x8x8 RUCs, however, the other RUC groups yielded similar outcomes. Observations and conclusions stated for the 8x8x8 are generalizable to the other RUC groups illustrated above.

ML architecture feature exploration

As GMC lacks any accurate predictive capability for shear stress and strain, the primary measure of interest presented in this work is the ML correction to shear strain and stress. Figure 5 shows the impact of the specification of dense layers and stride on overall slope of the idealized response vs GMC+ML response for shear strain. Similarly, Figure 6 shows the impact of the specification of dense layers and stride on overall slope of the idealized response vs GMC+ML response for shear strain. It is immediately apparent that all ML models improve the prediction of shear strain. The slope vs ideal goes from 0.003 (uncorrelated, essentially unpredictable by GMC method) to be greater than 0.635 for *all models*. However, it also becomes apparent that a convolutional stride of 1 is unable to make great improvements with any number of convolutional and dense layer filters (blue dots, all in the range of 0.635 to 0.704). Conversely, a stride of 2 or 3 yields significant additional improvement with most convolutional and dense layer specifications. Both yield maximum slopes that are significantly higher (stride 2 max = 0.916, stride 3 max = 0.970).

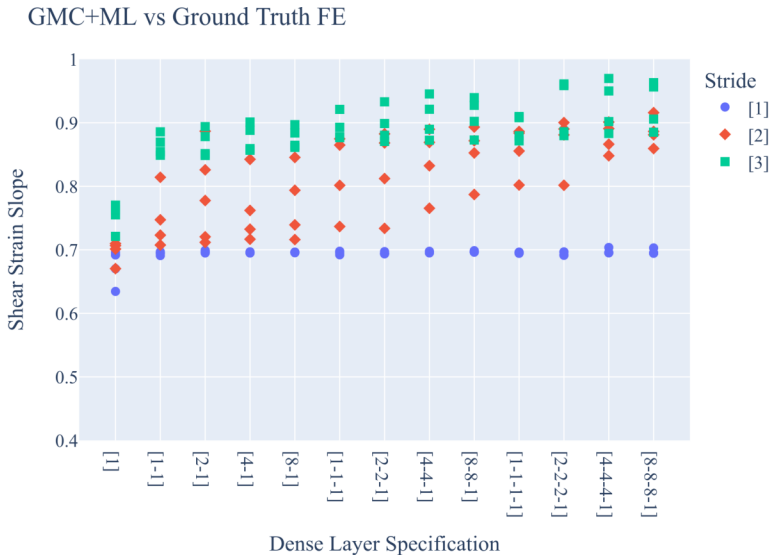


Figure 5: Shear strain slope for GMC+ML vs Ground Truth FEA, as a function of Dense Layer Specification

Further examination of Figure 6 demonstrates that having two convolutional layers (i.e., specs [1-1], [2-2], [4-4], and [8-8] with a stride of 3 produces a good prediction (slopes ranging from 0.849 to 0.901). Further, additional convolutional layers beyond 2 only provides a modest improvement in slopes *for stride = 3*. However, increasing the number of convolution and dense layers (and/or the number of neurons per layer) is helpful when the convolutional stride = 2 (slopes trending from ~0.7 to ~0.92) with increasing model complexity.

It is useful to consider the cost of using the ML model once trained. This scales with the number of model parameters. Convolutional and dense layers do not have the same cost in terms of traininable parameters. Further, for padding=”same” as used in this work, increasing the stride also significantly increases the number of trainable parameters. The cost of the models are examined in Figure 7 and Figure 8. These show the parameter cost of increasing stride to 3 is substantial, and that dense layers can be used for comparatively lower cost at lower stride levels. Further, it shows that increasing the number of convolutional layers and the number of filters per layer also increases parameter cost substantially.

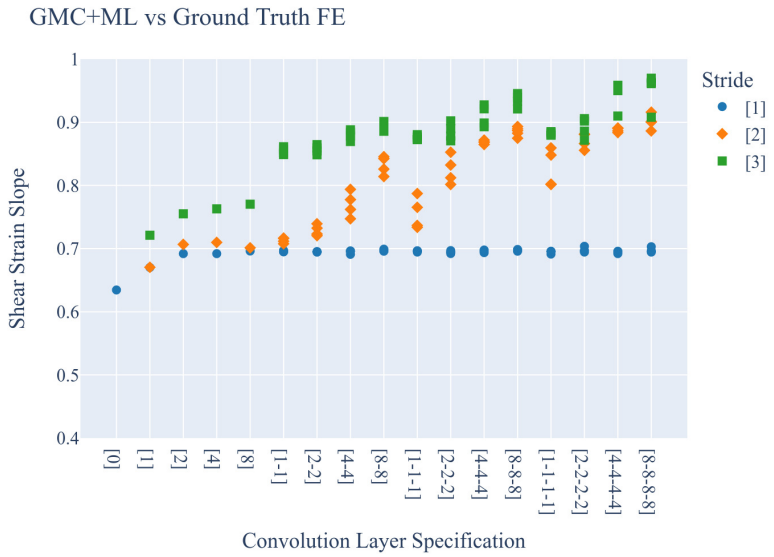


Figure 6: Shear strain slope for GMC+ML vs Ground Truth FEA, as a function of Dense Layer Specification

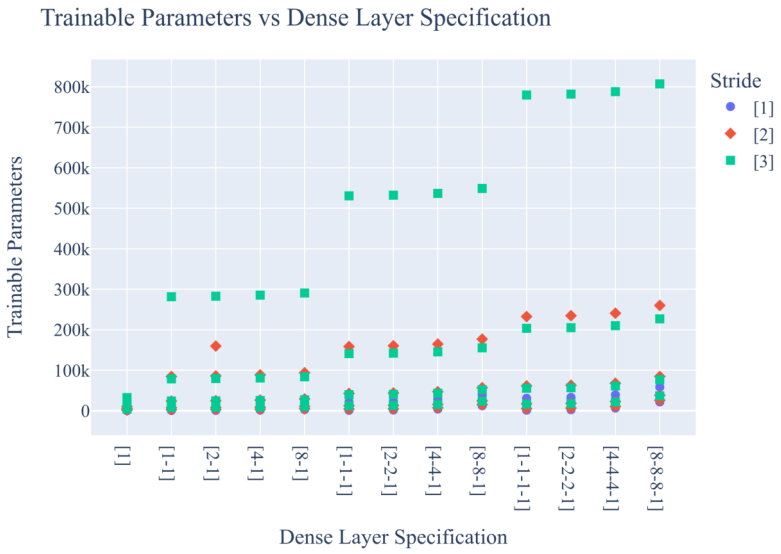


Figure 7: Trainable Parameters as a function of Dense Layer Specification

Figure 9 plots the number of trainable parameters vs slope, and is meant to convey a higher quality subset of the ML models (where slope > 0.875, r^2 > 0.875). These models are down-selected to include *only those models that have less than 50,000 trainable parameters*. The figure suggests models that would be effective at quickly correcting GMC models *to more accurately predict shear strain*. At the far right is an accurate "low cost" model, henceforth called a fast-model. It has three convolutional layers [2-2-2] (which, per the specifications described above, applies 2x12 = 24 convolutional filters per layer) and three dense layers [4-4-1] (which is 4x12=48 neurons per layer, except for the last layer). It uses a stride of 3. The total number of trainable parameters is 43,092. A second high quality model is suggested at the bottom of Figure 9. That model has four convolution layers [1-1-1-1] with 1x12=12 filters per layer. It has four dense layers [2-2-2-1] at 2x12=24 neurons per layer (except the last layer). It also uses a stride of 3. The total number of trainable parameters is 17,412. It is slightly less accurate in terms of ideal slope but at approximately 40% the cost of use. This is also referred to as a fast model.

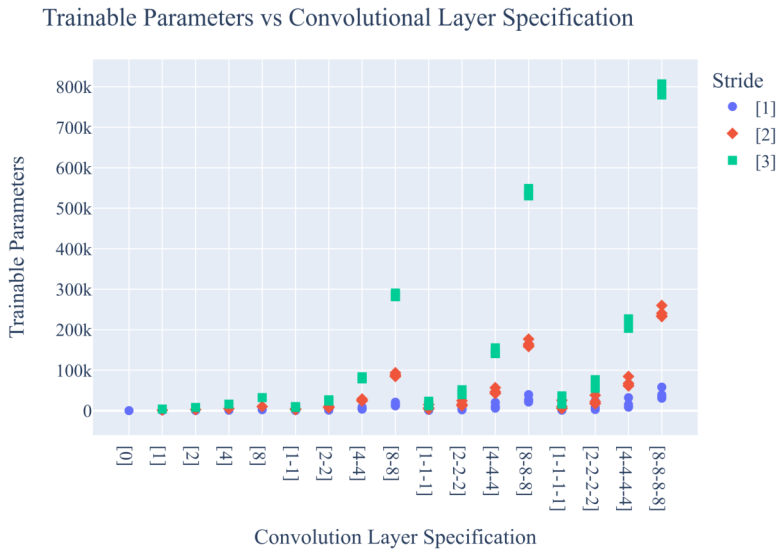


Figure 8: Trainable Parameters as a function of Convolutional Layer Specification

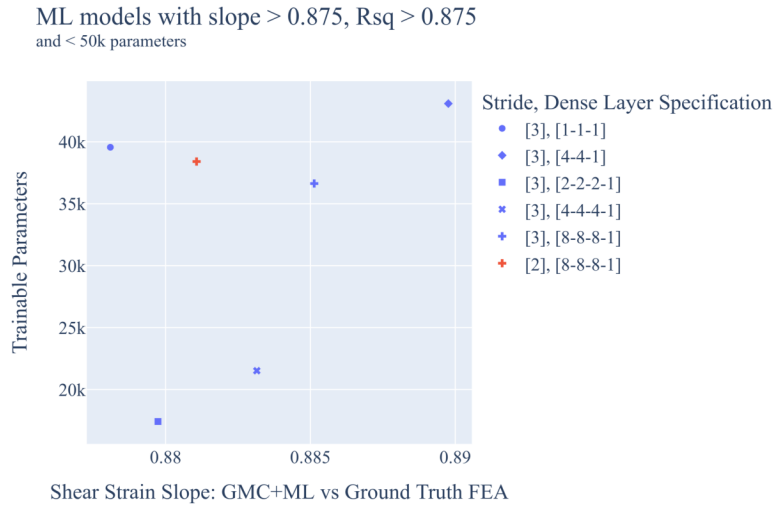


Figure 9: Fast and Accurate Model Predictions

As described in Figures 5 - 9, the grid search method of optimization yielded many ML models with varying predictive accuracy and costs. Predictions of stress and strain from notable architectures are presented in the following subsections and Figures 10 - 13. Two are the selected fast models described in the preceding paragraph. Two other models are examined in the subsequent subsections: a model that produces the highest slope (nearest slope to the ideal ground-truth line) and a model that produces the highest Pearson Correlation Coefficient (r^2 value, i.e., it has the least "noise" in its prediction while still providing a prediction near the ideal ground truth line). All are split into normal and shear components of stress and strain over 4 figures. The scatterplots represent the quantity for each subcell for all models of the testing dataset. The size of the test datasets were 600 GMC models (1000 models x 12 steps with 1/20th reserved for testing). Each regression represents 600x8x8x8x3 individual subcell outcomes. (Visualization is limited to 250k points per regression).

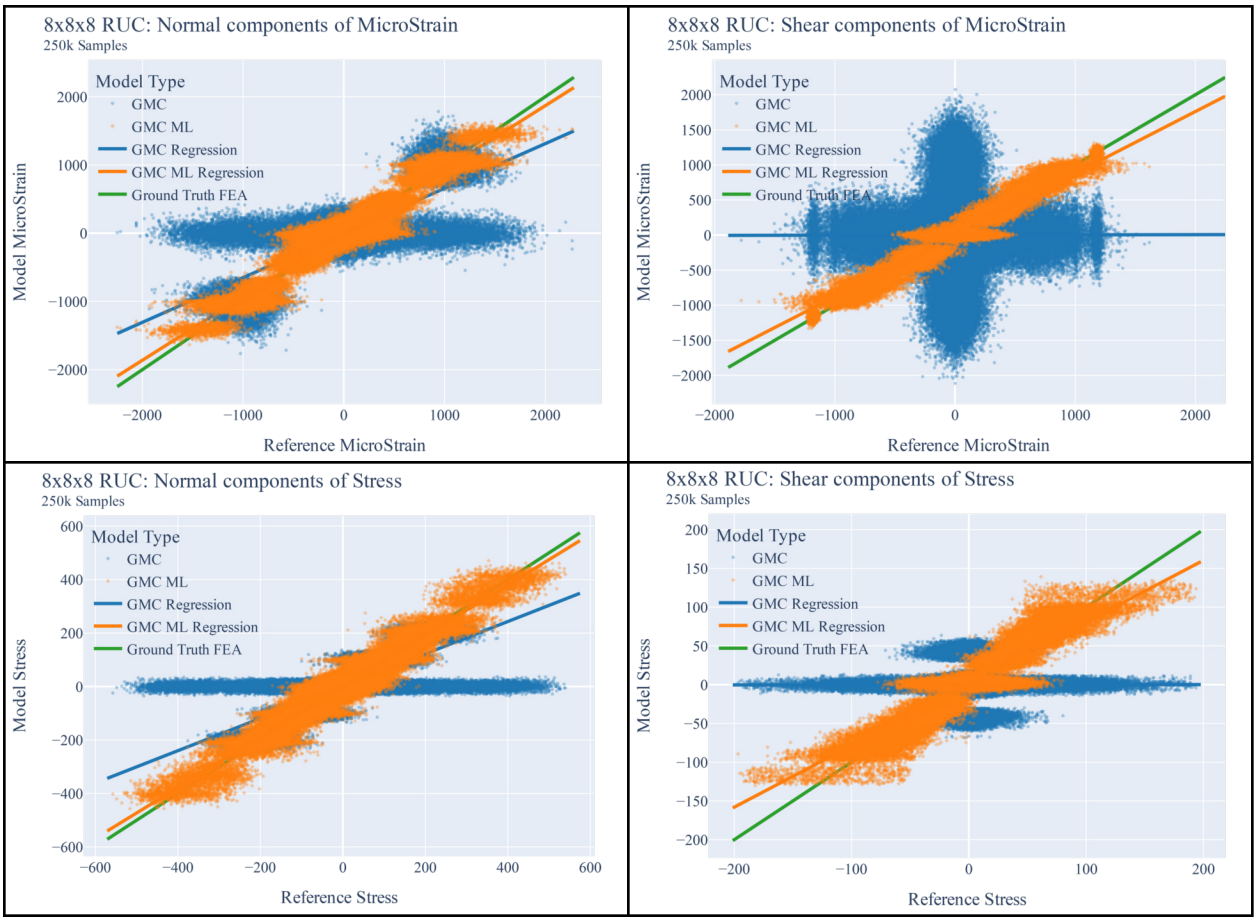
Fast Model: Predictions with 4 Convolutional Layers [1-1-1-1], 4 dense layers [2-2-2-1], and stride=3 (17,412 parameter model)

Figure 10 shows normal strain, shear strain, normal stress, and shear stress values aggregated over the models and into RUC subcells for the ML models with 4 convolutional and 4 dense layers, with a stride of 3 (as described in the subheading). GMC, GMC+ML, as well as regressions and an ideal reference line are plotted against ground

truth for each corresponding subcell within the models. The ML model results shown are meant to demonstrate a fast ML model with a comparatively small number of parameters.

Examining first GMC results alone, it is apparent that GMC normal strains reproduce in aggregate the applied normal strain in the direction of loading (i.e., the clouds of points nearing ±1000 microstrain on the ideal line) whereas the transverse strains are less accurately captured (the predominantly horizontal cloud of points along the model microstrain ~=0). After enhancement with ML, the horizontal cloud transitions to be more closely aligned with the ideal slope. This means the transverse (i.e., Poisson) strains are being more accurately captured. Further, the normal strain concentrations near grain boundaries are also more accurately captured after enhancement (shown by the greater alignment of the cloud of points near 1000 microstrain to be more along the reference line). The normal stresses exhibit very similar outcomes, although with slightly different slope and r^2 for the corresponding linear regressions.

More significant than normal strain and stress are the shear strain and stress. Examining only the GMC results for shear strain, the GMC models produce a "+" shaped cloud of points when plotted against the ground truth FEA shear strain. This reflects shear strain being predicted where no shear strain should exist (the vertical aspect of the "+" shaped cloud) and also no shear strain being predicted where shear strain should exist (the horizontal aspect of the "+" shaped cloud). For GMC, the regression yields a slope=0.003 and a Pearson Correlation Coefficient r^2=0.000 Hence, shear strain is inaccurate and uncorrelated in these GMC models. This is a known limitation of GMC. Upon ML enhancement, the cloud of points predominately follows the idealized reference line (slope=0.878, r^2=0.876). A similar outcome is observed in the plot of shear stress, both for GMC alone and for GMC+ML models. The ML enhanced shear stresses are observed to have slope=0.784 and r^2=0.796.



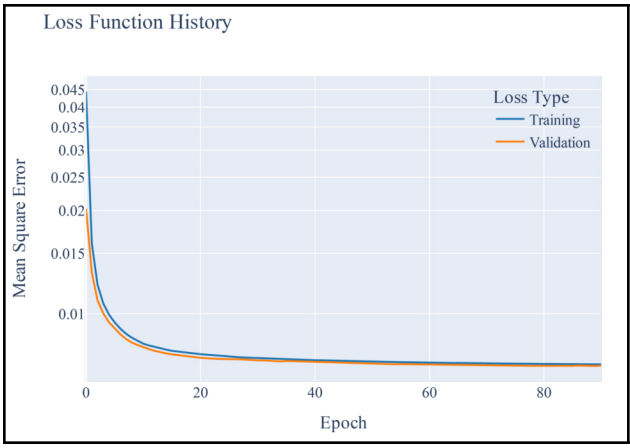
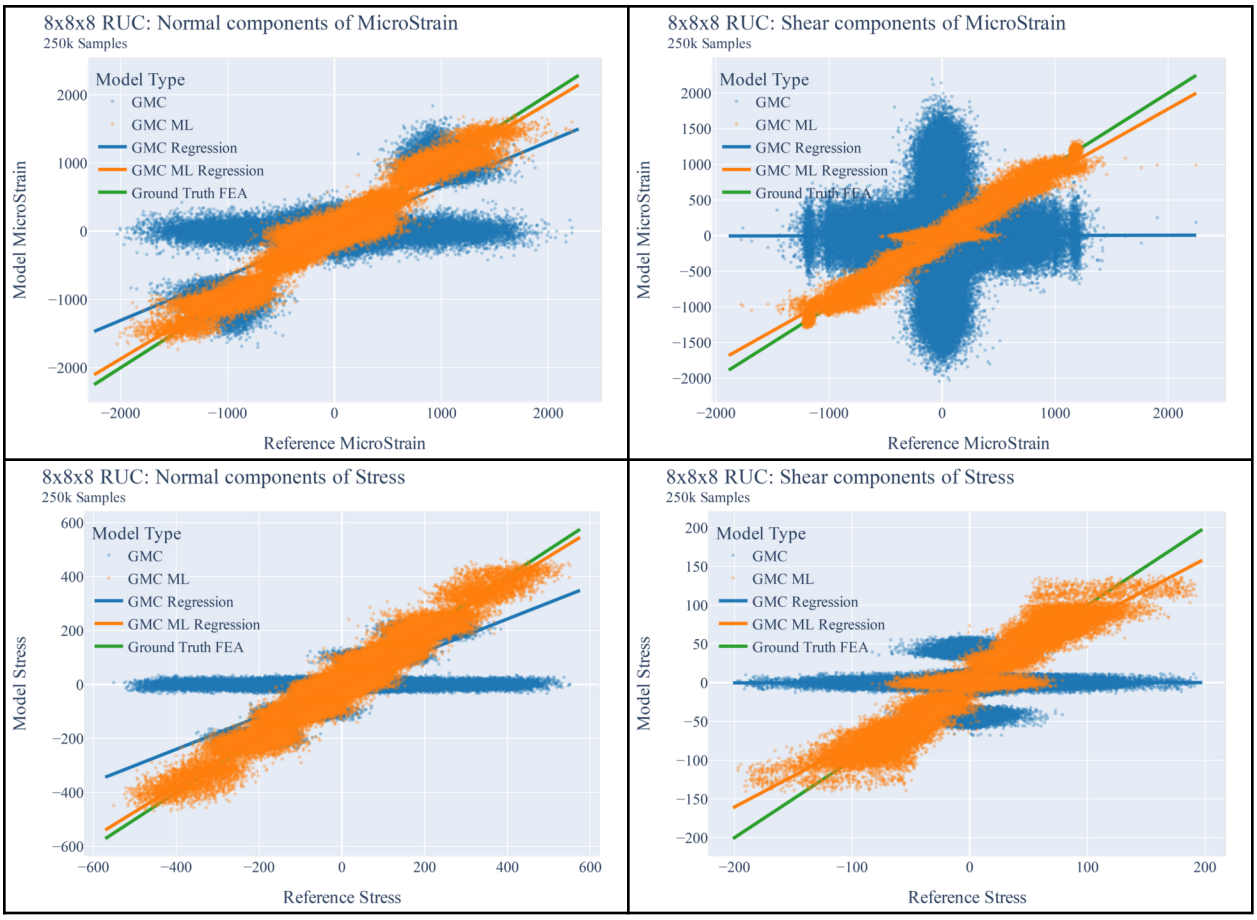


Figure 10: Normal strain, shear strain, normal stress, and shear stress values aggregated over the models and into RUC subcells for the ML models with 4 convolutional and 4 dense layers, with a stride of 3. Also, training loss for this architecture.

Fast Model: Predictions with 3 Convolutional Layers [2-2-2], 3 dense layers [4-4-1], and stride=3 (43,092 parameter model)

Figure 11 shows normal strain, shear strain, normal stress, and shear stress values aggregated over the models and into RUC subcells for the ML models with 3 convolutional and 3 dense layers, with a stride of 3 (as described in the subheading). These figures represent a second fast model, slower than the first reported, but nevertheless with a modest number of model parameters.



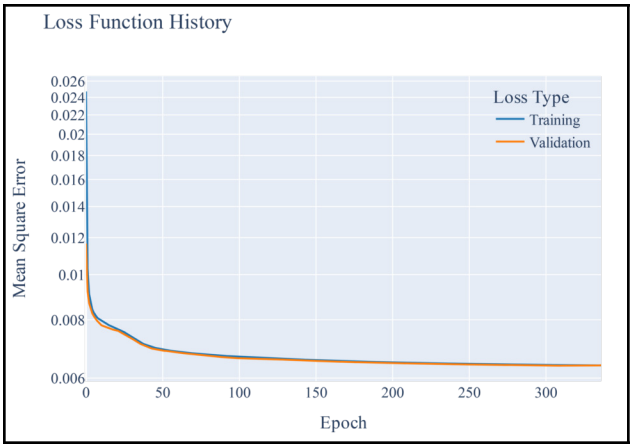
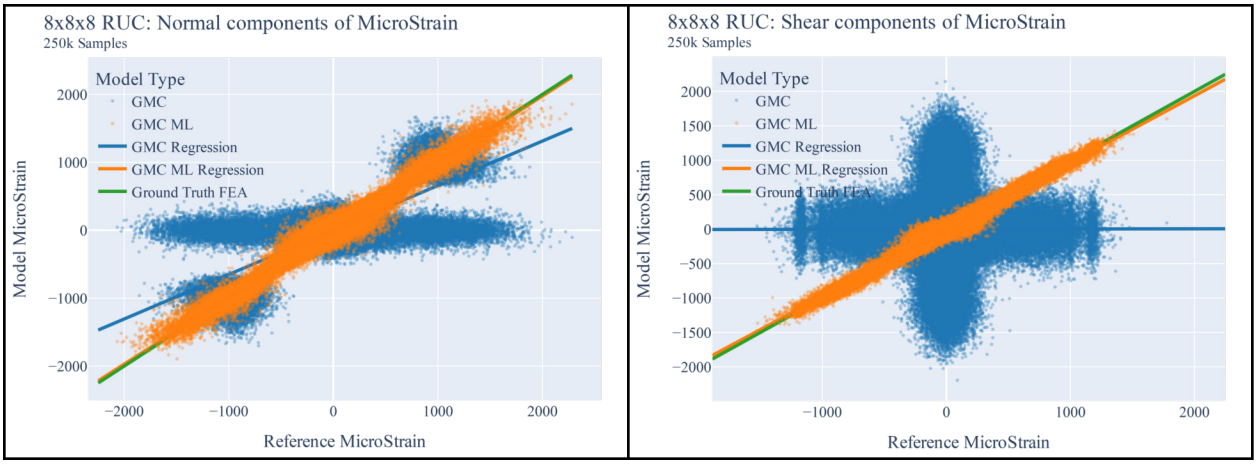


Figure 11: Normal strain, shear strain, normal stress, and shear stress values aggregated over the models and into RUC subcells for the ML models with 3 convolutional and 3 dense layers, with a stride of 3. Also, training loss for this architecture.

In Figures 10 - 13, the GMC model results are identical, hence, the descriptions in the prior section apply to these figures as well. The ML model described herein behaves quite similarly: the cloud of GMC+ML enhanced points for shear and normal predominately follows the idealized reference line. The regression statistics differ slightly and are slightly improved (slope=0.890, r^2=0.882). The ML enhanced shear stresses are observed to have slope=0.782 and r^2=0.800.

Best Shear Strain Matching: Predictions with 4 Convolutional Layers [8-8-8-8], 4 dense layers [4-4-4-1], and stride=3 (787,932 parameter model)

Figure 12 shows normal strain, shear strain, normal stress, and shear stress values aggregated over the models and into RUC subcells for the ML model which produced the best slope outcome for shear strain (slope=0.970, r^2=0.964) and shear stress (slope=0.788, r^2=0.774). These figures represent slower models with significantly more model parameters. Nevertheless, they provide insight into the limits of accuracy of enhancement.



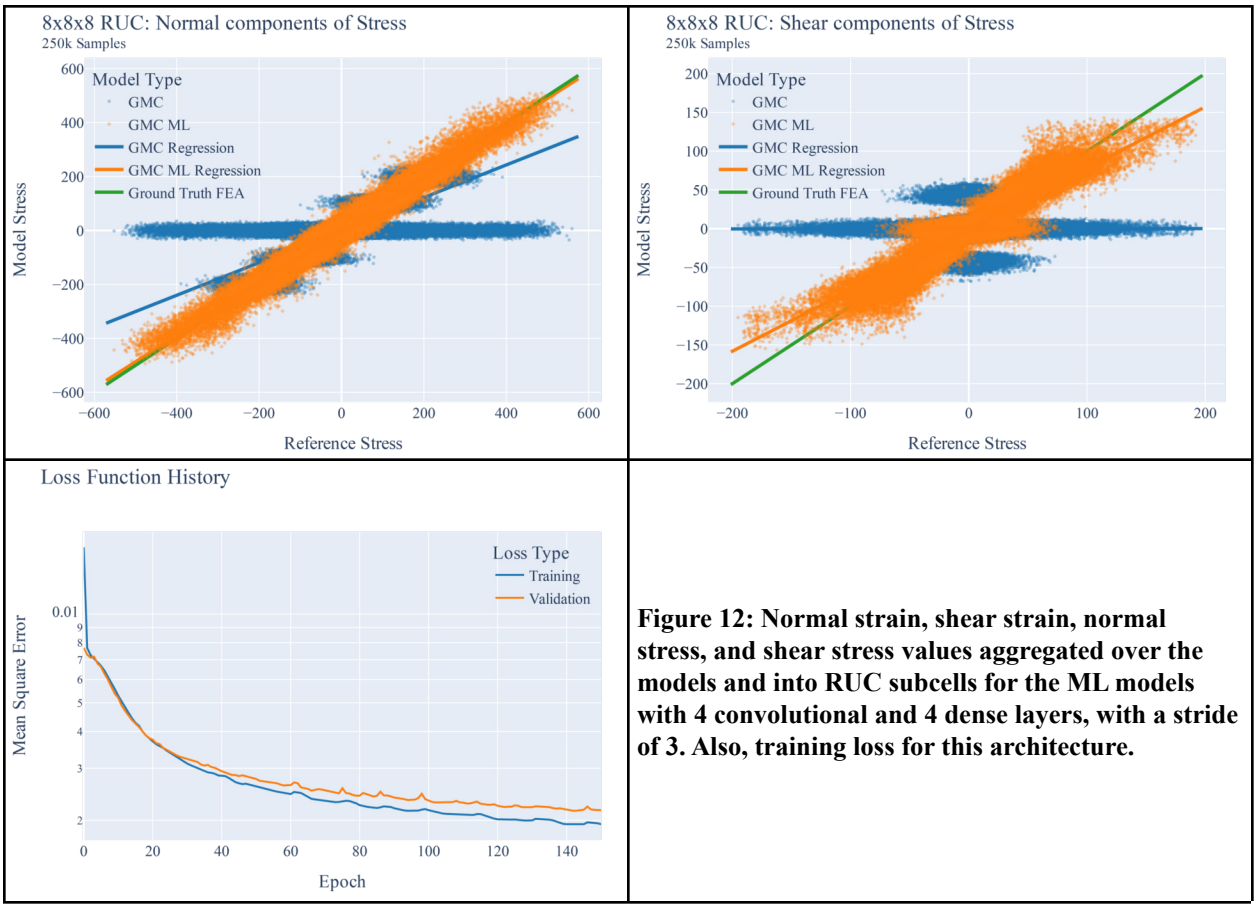
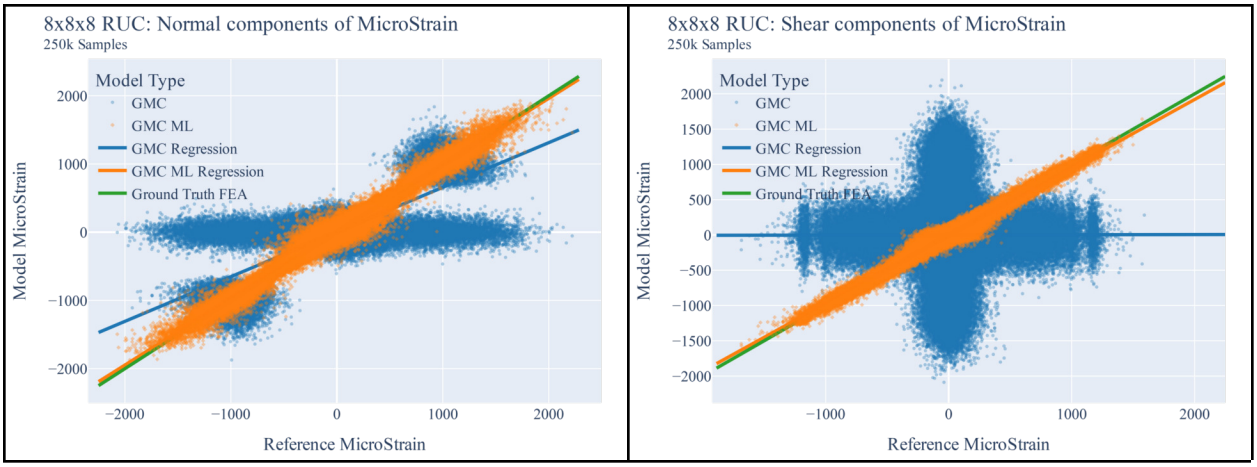


Figure 12: Normal strain, shear strain, normal stress, and shear stress values aggregated over the models and into RUC subcells for the ML models with 4 convolutional and 4 dense layers, with a stride of 3. Also, training loss for this architecture.

Best Pearson Correlation Coefficient (r^2) for Shear Strain: Predictions with 4 Convolutional Layers [8-8-8-8], 4 dense layers [8-8-8-1], and stride=3 (807,084 parameter model)

Figure 13 shows normal strain, shear strain, normal stress, and shear stress values aggregated over the models and into RUC subcells for the ML model which produced the best slope outcome for shear strain (slope=0.963, r^2=0.968) and shear stress (slope=0.792, r^2=0.789). These figures represent slower models with significantly more model parameters. Nevertheless, they provide insight into the limits of accuracy of enhancement.



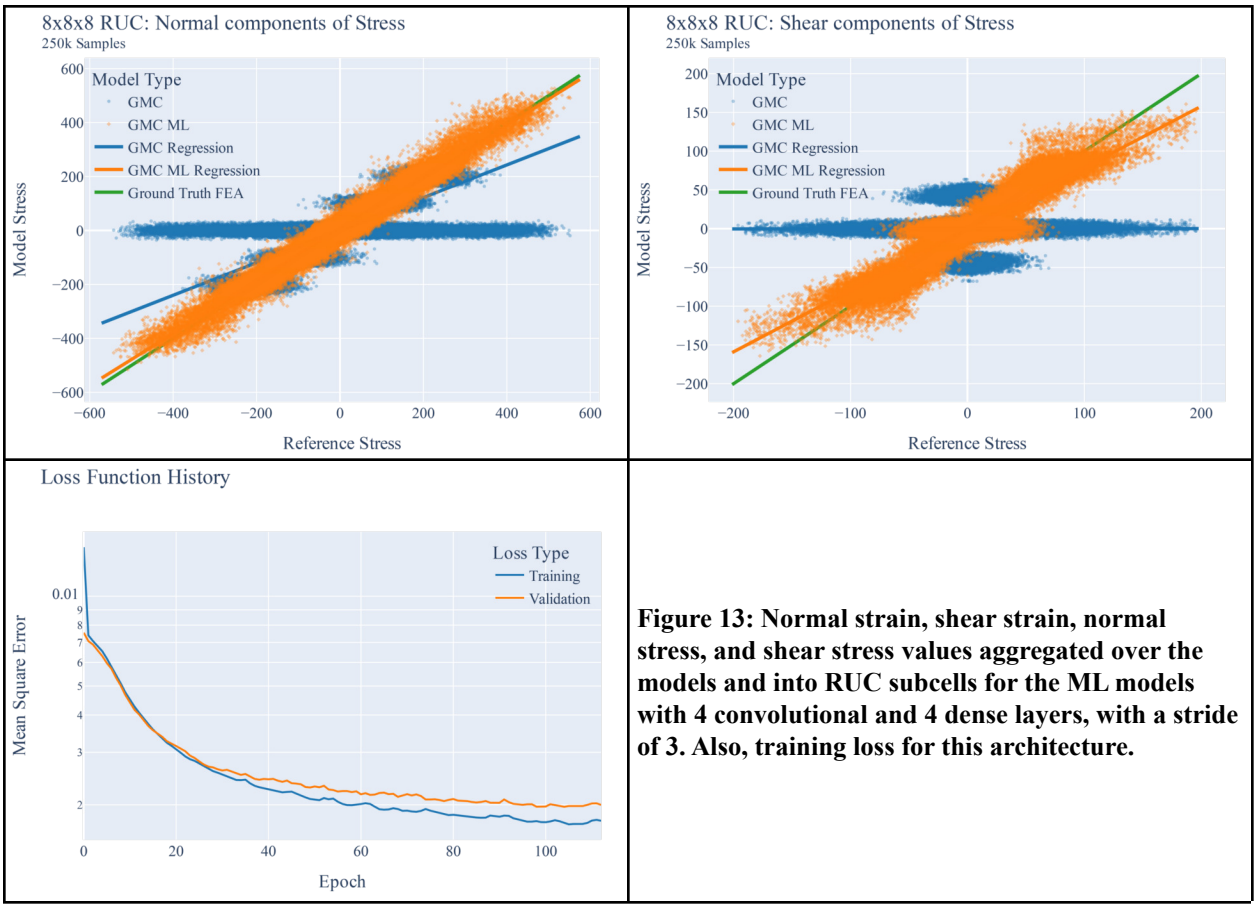


Figure 13: Normal strain, shear strain, normal stress, and shear stress values aggregated over the models and into RUC subcells for the ML models with 4 convolutional and 4 dense layers, with a stride of 3. Also, training loss for this architecture.

Speed of correction:

The intent of the ML correction is to improve the predicted strains and stresses relative to GMC alone and to do so at low cost relative to the alternatives such as HFGMC. Table 1 shows the computational cost of an 8x8x8 RUC FE, GMC, and ML corrections. The timing is reported for single CPU solutions (FE and GMC models) using an Intel (R) Xeon(R) W-2145 CPU @ 3.70GHz. For ML corrections, the timing is reported as total time to calculate all model corrections divided by the number of corrections. The ML correction speeds were enhanced using parallel predictions on a NVIDIA TU104GL Quadro RTX 4000 GPU. Parallel ML predictions are reported because this is reflective of a likely multi-scale FE modeling use-case where GMC+ML is used as the constitutive model at every integration point of a FE model. MAC/GMC is used to establish the reference time. MAC/GMC and NASMAT GMC produce identical results, however, the NASMAT research code has not yet been optimized. Hence, MAC/GMC provides a superior reference for computational cost.

Table 1: Computational cost. Correction cost % is defined as the ML correction time divided by the GMC single CPU, single load model time (Run with MAC/GMC).

Method	Convolutional Layer Specification	Dense Layer Specification	Stride	Calculation Time (milliseconds)	Correction Cost %

GMC: Single CPU, Single Load (Optimized MAC/GMC)	NA	NA	NA	**27.00**	NA
GMC: Single CPU, Single Load (Unoptimized NASMAT)	NA	NA	NA	603.00	NA
CalculiX: Single CPU, Single Load (Hex Mesh)	NA	NA	NA	374.24	NA
CalculiX: Single CPU, Single Load (Tet Mesh)	NA	NA	NA	708.81	NA
ML Correction: Multiple corrections, GPU parallelized	[1-1-1-1]	[2-2-2-1]	3	1.84	6.8%
ML Correction: Multiple corrections, GPU parallelized	[2-2-2]	[4-4-1]	3	1.40	5.2%
ML Correction: Multiple corrections, GPU parallelized	[8-8-8-8]	[4-4-4-1]	3	10.58	39.2%
ML Correction: Multiple corrections, GPU parallelized	[8-8-8-8]	[8-8-8-1]	3	11.66	43.2%

Overfit and underfit

Successful training of a neural network exhibits a decreasing loss function for both training and validation loss as training proceeds. In each of the two fast model results presented above, the predictions are followed by a training history plot showing loss as a function of epoch. In these figures, the training and validation curves decrease and converge asymptotically and with validation loss near training loss. It can thus be concluded from the training, validation, and test loss outcomes that the trained models appear to be accurate, well behaving, and do not exhibit overfitting. However, in each of the models with a large number of parameters (i.e., the best slope and best Pearson Correlation Coefficient), the plot of training history shows that the validation loss is somewhat larger than the training loss after approximately 20 epochs. Nevertheless, the validation loss continues to decrease with additional training. This suggests that the large parameter models are likely accurate and improving with training, however, overfitting is a concern and must be monitored carefully.

V. Discussion

Homogenization vs localization

Multiple prior studies have shown GMC to be an effective tool for predicting homogenized response. Further, most prior use of GMC has been evaluated with an eye to accurate reproduction of macroscopic behavior *after homogenization*. In contrast, the results described in this work emphasize localized prediction of stress and strain at the subcell level *without attention to homogenization*. While ML may be useful for improving the homogenization process, no effort has been made to assess whether ML homogenization would match GMC or FEA homogenization outcomes. This article anticipates future hierarchical multi-scale models where the localization drives a lower level model. When GMC is intended for use as an intermediate scale, the accuracy of GMC (or GMC+ML) localization

will factor prominently into the loads evaluated at the lowest scales. Accurate shear stresses and strains are likely necessary for assessment of localized damage mechanics to the lowest scales.

Speed and computational cost

Direct comparison of computational cost among the methods (FEA, GMC, ML; reported in Table 1) has only been examined for one RUC type (8x8x8). NASMAT is a major refactoring of its predecessor (MAC/GMC) with a current focus on parallelism and a hierarchical call structure. Reported timings for CalculiX and MAC/GMC are based on production code with a single CPU (MAC/GMC was used only to develop a reference solution cost and not for ML training).

The NASMAT and ML costs reported must be considered approximate and are not necessarily representative of the costs in a production environment. Neither the NASMAT GMC results nor the ML corrections report timings that are based on code that has been developed to optimize single RUC analysis efficiency. The current state of the NASMAT is one of implementation and feature addition rather than speed optimization. Timings for NASMAT (and MAC/GMC) single load and single CPU models may not be reflective of the costs of running many parallel NASMAT model outcomes in a multi-scale FE model. By contrast, NASMAT in parallel as a FE constitutive model permits some overhead to be amortized over many instances of the function calls. The comparison of average ML correction cost (GPU enhanced) to single model NASMAT models (CPU only) is neither like-kind nor well reflective of production analysis intent.

Additional exploration of ML architectures and further hyperparameter optimization is anticipated to decrease the cost of use of the ML approach, perhaps with increases in accuracy. Also, average cost of ML predictions depend on how many predictions are done. The cost of running 1 or many predictions on the GPU are similar up to the number of predictions that can fit in memory on the GPU. *Hence, all model timings reported in this work differ relative to the cost of implementation if a NASMAT GMC+ML model were used as the constitutive model for a multi-scale FE model (i.e., the intended use case).* However, the results do represent a first attempt to quantify the intended use case given the current state of the tools.

Despite the limitations associated with the reported computational costs, this work demonstrates that GMC models combined with a machine learning surrogate can inexpensively produce localization that is superior to GMC alone. Based on the timings in Table 1, the cost of NASMAT GMC is anticipated to be less than 10% of a comparable FE submodel surrogate. The cost of improving the results via ML should be significantly less than 10% of the cost of the GMC model calculation.

Anticipated alternative ML enhancements

This work examined training ML models for only one level of RUC subcell (i.e., GMC and FEA). Machine learning was used to train models to use the outputs of low fidelity but fast models to predict outputs of models of higher fidelity at the same number of subcells. However, another aspect of model fidelity is the fidelity of the RUC to the physical grain structure. This latter aspect was only partially explored in having an increasing number of subcells (i.e., n = {2, 4, 8, 16, 32}) in the grain structure representation, but in isolated evaluations. No comparison between these levels of subcells is reported herein.

It is anticipated that another ML approach may be able to do model enhancement that functionally increases the number of subcells in an RUC. This is similar to the machine learning based super resolution techniques used to upscale photographic and other images. Though conceptually possible, the utility of a ML structural super-resolution technique for enhancement of damage progression model accuracy is currently unknown. Ongoing work includes the training of machine learning models which promote the size of the RUC (i.e., from n=8 to n=16).

Limitations

The machine learning surrogates are trained only on randomly generated metallic microstructures with one constitutive description and random grain size. The work has not explored multiple grains of different elastic moduli nor other material systems such as fibrous composite materials. It is likely that, for the process described above to be generalizable across all material systems, additional GMC and FEA models of different material systems must be included into the training set.

Despite the limitations noted, this work clearly demonstrates that fidelity improvement (or fidelity correction) is possible using machine learning methods for structural micromechanics models. This technique could be used to address one of the most significant criticisms of the generalized method of cells and its related methods, that is, the lack of accuracy and precision for the localization of stress and strain.

VI. Conclusion

This article has demonstrated that a generalized method of cells model of metallic microstructures can be enhanced using a convolutional neural network to produce localized (subcell level) strains and stresses that are much closer to ground truth finite element models than a generalized method of cells model alone with significantly increased computational efficiency compared to finite element models. Shear strain and stress prediction are not accurate in the generalized method of cells. Machine learning enhanced shear strains, measured in terms of the linear regression coefficients as a function of ground truth strains, were enhanced from inaccurate and uncorrelated (slope=0.003, r^2=0.000) to accurate and well correlated (slope=0.890, r^2=0.882) relative to ground truth (Ideal slope =1.0, r^2 = 1.0). Enhanced shear stresses had a similar accurate and well correlated result (slope=0.782, r^2=0.800). In applying the convolutional neural network, a convolutional stride of 1.0 (padding='same') was only modestly effective while strides of 2 or 3 were more effective yet at higher cost. Additional convolutional layers were generally more expensive than additional dense layers, often with limited benefit. The accuracy of enhanced localized shear strains and stress is expected to yield benefits for damage progression models, especially in the context of hierarchical multi-scale methods where the generalized method of cells is applied at the intermediate scale.

Acknowledgments

This work was funded via a NASA summer faculty fellowship and a NASA visiting scholar fellowship. Additional funding was from the Western Michigan University Sabbatical Program. NASA funding was provided by the Transformational Tools and Technologies (TTT) project, under the Aerospace Research Mission Directorate. These funding sources are gratefully acknowledged.

References

[1] J. Allison, D. Backman, and L. Christodoulou, "Integrated computational materials engineering: a new paradigm for the global materials profession," *Jom*, vol. 58, no. 11, pp. 25–27, 2006.

[2] J. H. Panchal, S. R. Kalidindi, and D. L. McDowell, "Key computational modeling issues in integrated computational materials engineering," *Computer-Aided Design*, vol. 45, no. 1, pp. 4–25, 2013.

[3] J. Allison, "Integrated computational materials engineering: A perspective on progress and future steps," *Jom*, vol. 63, no. 4, p. 15, 2011.

[4] N. R. Council and others, *Integrated computational materials engineering: a transformational discipline for improved competitiveness and national security*. National Academies Press, 2008.

[5] "Stanford Engineering Everywhere | CS229 - Machine Learning." https://see.stanford.edu/course/cs229 (accessed Nov. 29, 2021).

[6] E. Alpaydin, *Introduction to machine learning*. MIT press, 2020.

[7] A. Sorini, E. Pineda, J. Stuckner, and P. A. Gustafson, "Use of a convolutional neural network for enhancing multi-scale modeling of composite materials," 2021.

[8] J. Aboudi, "The Generalized Method of Cells and High-Fidelity Generalized Method of Cells Micromechanical Models—A Review," *Mechanics of Advanced Materials and Structures*, vol. 11, no. 4–5, pp. 329–366, 2004, doi: 10.1080/15376490490451543.

[9] T. M. Ricks, T. E. Lacy, E. J. Pineda, B. A. Bednarcyk, and S. M. Arnold, "Computationally efficient High-Fidelity Generalized Method of Cells micromechanics via order-reduction techniques," *Composite Structures*, vol. 156, pp. 2–9, Nov. 2016, doi: 10.1016/j.compstruct.2016.05.093.

[10] M. Ghorbani Moghaddam, A. Achuthan, B. A. Bednarcyk, S. M. Arnold, and E. J. Pineda, "A multiscale computational model combining a single crystal plasticity constitutive model with the generalized method of cells (GMC) for metallic polycrystals," *Materials*, vol. 9, no. 5, 2016.

[11] L. F. Silva, F. A. Y. Genao, E. J. Pineda, and P. A. Gustafson, "Evolving Material Porosity on an Additive Manufacturing Simulation with the Generalized Method of Cells," in *AIAA Scitech 2020 Forum*, American Institute of Aeronautics and Astronautics. doi: 10.2514/6.2020-1124.

[12] P. Naghipour *et al.*, "Multiscale static analysis of notched and unnotched laminates using the generalized

method of cells," *Journal of Composite Materials*, vol. 51, no. 10, pp. 1433–1454, May 2017, doi: 10.1177/0021998316651708.
[13] K. C. Liu, A. Chattopadhyay, B. Bednarcyk, and S. M. Arnold, "Efficient Multiscale Modeling Framework for Triaxially Braided Composites using Generalized Method of Cells," *Journal of Aerospace Engineering*, vol. 24, no. 2, pp. 162–169, Apr. 2011, doi: 10.1061/(ASCE)AS.1943-5525.0000009.
[14] J. Aboudi, S. M. Arnold, and B. A. Bednarcyk, *Micromechanics of composite materials: a generalized multiscale analysis approach*. Butterworth-Heinemann, 2012.
[15] K. Balusu, T. Skinner, and A. Chattopadhyay, "An efficient implementation of the high-fidelity generalized method of cells for complex microstructures," *Computational Materials Science*, vol. 186, p. 110004, Jan. 2021, doi: 10.1016/j.commatsci.2020.110004.
[16] J. Aboudi, "Micromechanically Established Constitutive Equations for Multiphase Materials with Viscoelastic–Viscoplastic Phases," *Mech Time-Depend Mater*, vol. 9, no. 2, pp. 121–145, Sep. 2005, doi: 10.1007/s11043-005-1085-x.
[17] J. ABOUDI, "The Generalized Method of Cells and High-Fidelity Generalized Method of Cells Micromechanical Models—A Review," *Mechanics of Advanced Materials and Structures*, vol. 11, no. 4–5, pp. 329–366, Jul. 2004, doi: 10.1080/15376490490451543.
[18] K. C. Liu and A. Ghoshal, "Inherent symmetry and microstructure ambiguity in micromechanics," *Composite Structures*, vol. 108, pp. 311–318, 2014.
[19] "MicroStructPy - Microstructure Mesh Generation in Python," *MicroStructPy*. https://docs.microstructpy.org (accessed Nov. 29, 2021).
[20] "Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities." https://gmsh.info/ (accessed Nov. 29, 2021).
[21] G. Dhondt and K. Wittig, "CalculiX: A Three-Dimensional Structural Finite Element Program," 1998. www.calculix.de (accessed Nov. 29, 2021).
[22] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," 2010.