

A Convolutional Neural Network for Multiscale Modeling of Composite Materials

Alexander Sorini¹

Western Michigan University, Kalamazoo, MI, 49008, United States

Evan Pineda² and Joshua Stuckner³

NASA Glenn Research Center, Cleveland, OH, 44135, United States

and

Peter A. Gustafson⁴

Western Michigan University, Kalamazoo, MI, 49008, United States

An artificial convolutional neural network was created to efficiently mimic a micromechanics model, the High Fidelity Generalized Method of Cells, for use in multi-scale structural finite element analysis. The network was found to quickly and accurately replicate the stiffness predicted by the micromechanics model using a 2D image of an idealized representative volume element of a fiber/matrix microstructure. The long-term goal of this work is to efficiently apply multi-scale methods for predicting the damage progression of a composite structure.

Nomenclature

CNN	=	convolutional neural network
ANN	=	artificial neural network
NN	=	neural network
GMC	=	generalized method of cells
HFGMC	=	high fidelity generalized method of cells
RUC	=	repeating unit cell
RVE	=	representative volume element
FEM	=	finite element method

I. Introduction

A. Background

Multiscale modeling in the context of this work is modeling the response of a structure across multiple length scales. For instance, the entirety of the structure itself may be called the *macroscale*, while the constituent materials that make up the composite microstructures would be called the *microscale*. Multiscale structural analysis often involves embedding a micromechanics model within a finite element simulation to act as the constitutive model. Microstructure stiffness, damage, and its effects can be computed at the microscale, homogenized, and passed ‘up’ to the finite element simulation. The structural response is predicted at the macroscale with global strain, localized, and passed ‘down’ to the microscale model.

Machine learning is a subset of AI in which algorithms are trained on large datasets to perform tasks without being explicitly programmed to do so. One of the most useful machine learning techniques is the artificial neural network (ANN). Neural networks, in their simplest form, are made up of an input layer, an arbitrary number of hidden layers, and an output layer. Each layer consists of nodes which act to receive input from the previous layer, and depending on the magnitude of this input, either fire to the next layer or not. The connections between nodes have weights associated with them. *Learning*, in this context, is the process of updating these weights based on training data. Nonlinearity can be introduced into the neural network by selecting the appropriate *activation functions*, or the criteria

¹ Graduate Student, Mechanical and Aerospace Engineering, Glenn Research Center, Western Michigan University.

² Aerospace Research Engineer, Multi-scale Multi-physics Modeling, Glenn Research Center.

³ Materials Informatics Scientist, Multi-scale Multi-physics Modeling, Glenn Research Center.

⁴ Associate Professor, Mechanical and Aerospace Engineering, Western Michigan University.

for whether a node fires to the next layer or not. In many cases, machine learning models can act as a ‘black box’ which reproduces the output for a given input of a complex system but with minimal cost.

B. Motivation

With composite materials seeing increased application in the aerospace and automotive industries, the ability to model them effectively becomes increasingly important. Modeling composites can be challenging and computationally expensive compared to modeling their monolithic counterparts. Most multiscale methods utilize repeating unit cells (RUCs) that can represent arbitrary microstructures such as square or hexagonal packed microstructures. This simplified analysis is incomplete, however, as failure is driven by randomness in the microstructure. This means that for failure predictions, representative volume elements (RVEs) are needed. Using conventional multi-scale techniques, these models are computationally expensive to implement in the finite element method because the response must be evaluated at each integration point at each element. Hence, many millions of micromechanics computations may be required in a typical coupled multi-scale finite element analysis. For example, the micromechanics model used in this work, the High-Fidelity Generalized Method of Cells (HFGMC), provides accurate solutions and can be applied to an extremely wide variety of composite problems. However, HFGMC’s high accuracy comes at the cost of large computational expense due to its complexity. This cost is further compounded by nonlinear phenomena such as inelasticity or damage beginning to develop in the constituents. This large cost is a common theme among micromechanics models and may be part of the reason why multiscale methods are not widespread outside of the research domain, even though the methods themselves are powerful.

Using machine learning through an appropriately trained neural network can provide a substantial speed-up to these expensive, time-consuming computations. It is hypothesized that a well trained model can accurately reproduce a micro-mechanics model in a small fraction of the time required for HFGMC. This work demonstrates the ability of a convolutional neural network (CNN) surrogate model to predict the output of the high fidelity generalized method of cells at relatively low cost.

II. Methods

This section will detail the steps necessary to create and preprocess the data to be used for machine learning as well as creation, testing, and optimization, of the neural network architecture. Data was generated for this task using NASA’s Micromechanics Analysis Code with the Generalized Method of Cells (MAC/GMC) with High Fidelity GMC analyses selected. HFGMC assumes a second-order displacement field with respect to the local subcell coordinates, as opposed to GMC’s first order displacement assumption, enabling HFGMC to capture the normal-shear coupling which is not present in GMC analyses. HFGMC was selected because the randomness in the microstructures drives a strong normal-shear coupling.

A. Training Data: Creation

Constituent materials (fiber and matrix) are considered isotropic for this work. With this assumption, the inputs needed to run HFGMC simulations to compute stiffness are the Young’s Modulus, Poisson’s Ratio, and ultimate tensile strength of the composite constituents, and the microstructure geometry. To create a dataset representative of possible real-world problems to be solved, material properties were generated using pyppy [1] (a custom python library) within the ranges shown in Table 1. These material properties are written to an input file for MAC/GMC. Random microstructure data was then generated using custom code supplied by NASA [2] and modified to accept the previously created input files. The input files request the computation of a homogenized constitutive response from the microstructure and then subsequently stop execution of the program. Twelve-thousand input files and MAC/GMC model outputs were generated in total. This output file is parsed for the material constitutive properties and non-zero stiffness matrix components which are then saved for subsequent training of the neural network. This process was completed for 12,000 MAC/GMC simulations which therefore constitute the training and validation sets.

	Fiber	Matrix
Young’s Modulus (MPa)	4000-250000	2000-6000
Poisson’s Ratio	0.1-0.5	0.2-0.5
Ultimate Tensile Strength (MPa)	2000-5000	50-210

Table 1: Material property ranges.

B. Training Data: Preprocessing

The training set data pre-processing was done using standard Python scientific computing & machine learning libraries. Constitutive material properties (input) and stiffness matrix components (output) are loaded into a dataframe using the Pandas library. Microstructure data (input) for all simulations was imported as a list. Feature and target (input and output) data were scaled to values between 0 and 1 consistent with best practices for efficiency and numerical convergence. (If feature or target magnitudes vary by many orders of magnitude, as is the case with stress and strain, the training algorithm may struggle to update weights appropriately due to limitations with numeric precision.)

This scaled data was then split into training, validation, and testing data categories. Training data was used by the network while algorithmically updating the weights in its connections. The algorithm then validated after updating the weights at the end of each epoch. The purpose of each-epoch validation was to ensure that the model performed well on unseen data, i.e., that its learning was on track. Once the training concluded, the model was further validated (i.e., tested) on the testing data set: the portion of the data set unseen during the training phases. The justification for these three phases (training, validation, testing) is as follows. The training modifies the weights, the validation does not update the weights but is a reference point which guides the training, and lastly the testing phase evaluates network accuracy against previously unseen data.

C. Neural Network Implementation

A special type of neural network called a convolutional neural network was selected for this application. The CNN differs from a traditional ANN in that it accepts images as inputs, and through a series of convolutional and pooling layers, extracts the relevant features from an input image. 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 image classification tasks but can be modified to perform a regression based on an image input. Hence, the models used with HFGMC for creating the training set were converted into 2D images that represent the fiber/matrix microstructure.

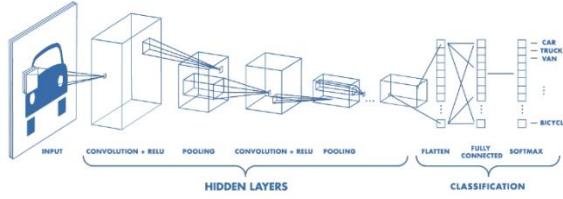


Figure 1.a: Convolutional neural network [3].

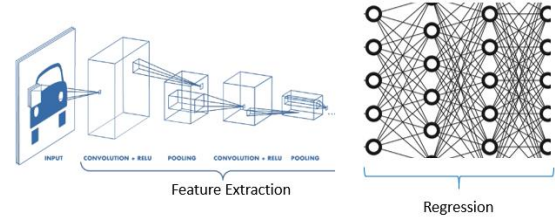


Figure 1.b: Convolutional neural network: repurposed [3].

These images were the model inputs to the CNN.

Figure 1.a and 1.b show a CNN to be used in classification, and a modified CNN to be used for a regression. To modify traditional the CNN architecture to be used in a regression, the layers near the end that perform the classification were removed. Only the layers which perform feature extraction were retained. The layers near the end of the network were replaced with a series of densely connected layers, and the output of the network was changed to be a layer of size 13, representing the 13 target variables the model is solving for. The activation function of the final layer was also changed from a SoftMax activation, which is typical for classification, to a linear activation, typical for regression.

To determine model performance, a loss function is required. The goal of any machine learning model is to minimize the value of this loss function through training. This happens as the models' predictions begin to match the dataset more closely. The loss function considered in this work is the mean squared error shown below, as is typical for regression problems.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

where:

- n = number of samples
- y_i = predicted output
- $\hat{y}_i$ = actual output

D. Network Optimization

With the general form of the network identified (the modified CNN for regression), the architecture optimization followed. A model-building function was defined in Python. Using a technique called Hyperband Optimization, a search space for model parameters was defined and swept over to find the models that produce the best results. Details of this search space can be seen in Table 2. The optimal values for these parameters are also shown. These optimal values were used to build the final network model, whose architecture is reported in the next section.

Table 2: Hyperband search space.

III. Results

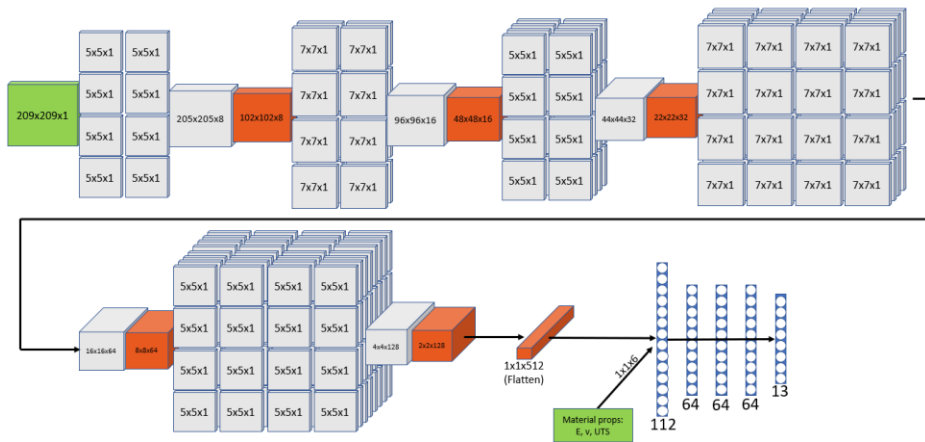


Figure 2: RVE_{Net} architecture.

B. Training, Validation, Testing

Fig. 3 aids visualization of the neural network training progress.

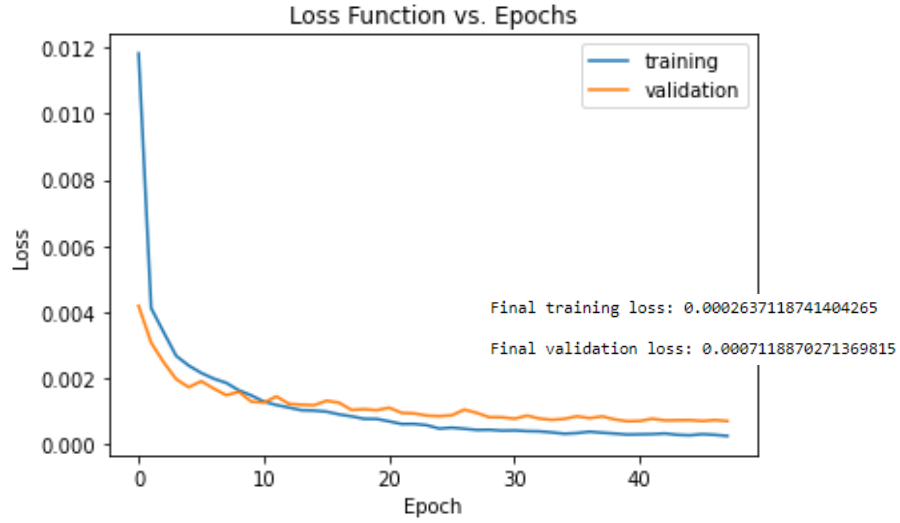


Figure 3: RVENet training progress.

The successful training of a neural network shows the loss function for both training and validation loss decreasing as training proceeds. It can be observed from Fig. 3 that the training and validation curves appear to settle asymptotically after rapidly improving over the initial epochs. This indicates the training has likely succeeded. Subsequent to training, the model was evaluated on the test data (the set of data it has not seen to train or to validate on). Listed in Fig. 4 is a short code snippet showing the evaluation of the model. The output of this block of code is the magnitude of the loss of the test dataset.

```
In [145]: 1 model.evaluate(
2           x={'image_input' : X_test_images, 'props_input' : X_test_props},
3           y=Y_test,batch_size=50)

48/48 [=====] - 0s 5ms/step - loss: 9.3297e-04

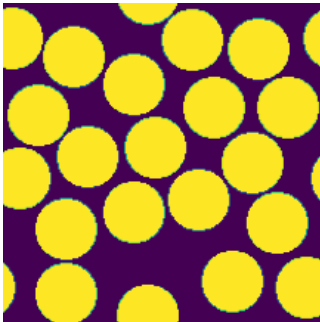
Out[145]: 0.0009329737513326108
```

Figure 4: RVENet testing results.

It can be seen from the model evaluation that the test data loss is slightly higher than the final validation loss, though still on the same order of magnitude. This suggests successful training.

C. Predictions

With the model trained and tested, it can thus be used to make explicit predictions on microstructures it has never seen before. Given the following microstructure and material properties, the predictions made by the network will be compared to the output computed using HFGMC:



	Fiber	Matrix
Young's Modulus (MPa)	66200	3459
Poisson's Ratio	0.35	0.48
Ultimate Tensile Strength (MPa)	3362	196

	Predicted	Actual	Difference	% Difference
C11	76053.03	76949.16	896.13	1.16
C12	44450.37	45579.19	1128.82	2.48
C13	42750	45549.92	2799.92	6.15
C14	-111.298	-81.1926	30.11	37.08
C22	60112.7	61096.19	983.50	1.61
C23	52425.23	50729.11	-1696.12	3.34
C24	-137.666	-67.4846	70.18	104.00
C33	61337.95	60975.35	-362.60	0.59
C34	-277.79	-267.689	10.10	3.77
C44	5158.701	4848.714	-309.99	6.39
C55	4097.712	4095.986	-1.73	0.04
C56	-134.971	-191.877	-56.91	29.66
C66	4059.653	4246.144	186.49	4.39
Computation Time	~.05sec/simulation	~22min/simulation	~~~~~	A lot

Table 3: RVENet predictions vs. HFGMC computations.

As can be seen from the prediction of this single microstructure, the network predicts C11 and C22 well. From this it is deduced that the model has learned the rule of mixtures and inverse rule of mixtures for these two components. Further, the model predicts well on most components with a few exceptions namely: C14, C24, C56. In predictions on other microstructural images, the model had trouble predicting C34 as well. Though this seems superficially problematic, upon deeper inspection it was determined that these coupling terms are reasonable for the small number of fibers in the RVEs used to train this network. The small number of fibers do not represent the statistics of larger microstructures and thus shouldn't be expected to have zero shear-normal coupling terms. The coupling terms will only be zero in an average sense (for a well-trained model). Conversely, if enough fibers are present (and enough models are used for training), the law of large numbers would drive these coupling terms towards zero and the stiffness matrix would be effectively transversely isotropic consistent with the macroscopic behavior of a composite ply. Since the network is for RVEs with a small number of fibers, any one RVE is expected to have non-zero coupling terms but the average should be zero if a large number of RVEs is evaluated. This was confirmed to be true for the trained network. The average of these components, as shown below, were all near-zero:

The average values over all the RVEs for the normal-normal and shear-shear constants are printed as well, demonstrating that they did not converge to zero (which they should not).

The average value of C14 is: -0.78
The average value of C24 is: 0.912
The average value of C34 is: 0.179
The average value of C56 is: -2.15

The average value of C11 is: 87431.19
The average value of C12 is: 16168.88
The average value of C13 is: 16175.67
The average value of C22 is: 27778.34
The average value of C23 is: 17157.84
The average value of C33 is: 27799.20
The average value of C44 is: 5241.540
The average value of C55 is: 5320.360
The average value of C66 is: 5313.978

D. Layer Activations

To visualize what is going on inside each layer of the trained model when it receives an input, the Keract package in Python was used. Only the activations of the first two convolution-pooling blocks are shown. Figure 5 shows the input image that was fed into the network. Figure 6.a and 6.b represent the activations of the first convolutional layer and the first pooling layer, respectively, and similarly for Fig. 7.a and 7.b representing the second layer: In the figures below, the details that the model identified can be observed. For instance, in the top center of Fig 6.a, the filter has learned to detect rounded edges. These first steps are just the beginning of feature extraction applied by the trained model. The subsequent pattern detection is difficult to understand and to represent graphically. The 'black box' nature of machine learning remains a barrier to confidence and adoption of these methods.



Figure 5: Image input.

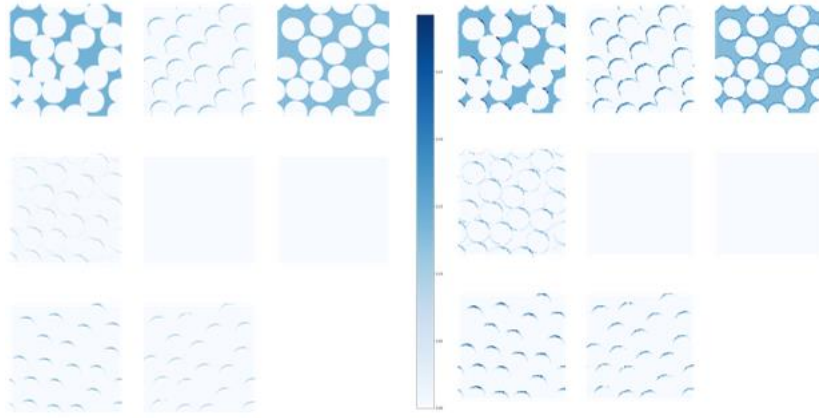


Figure 6.a: 1st Convolutional Layer.

Figure 6.b: 1st Pooling Layer.

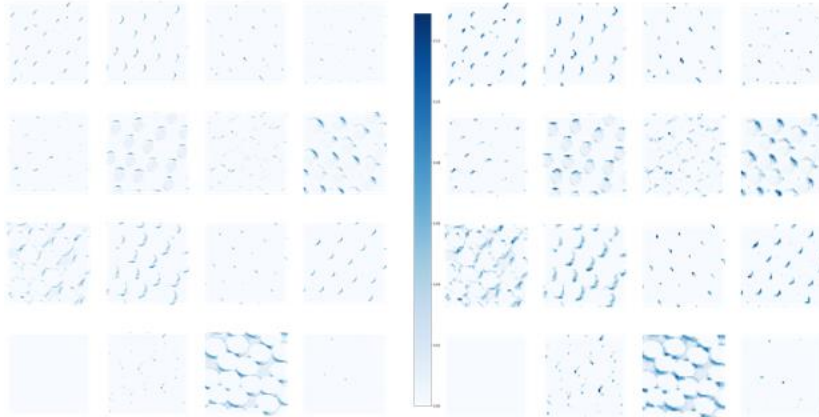


Figure 7.a: 2nd Convolutional Layer.

Figure 7.b: 2nd Pooling Layer.

IV. Conclusion

A methodology has been demonstrated for training a convolutional neural network to quickly replicate a stiffness prediction from an image of a 2D HFGMC micromechanics model. The model represents a low-cost, high-speed alternative for stiffness computation in multiscale analyses. The neural network model was found to perform approximately 25,000 times faster than its physics-based counterpart on isolated, small representative volume elements where micromechanical initial stiffness prediction is desired. Performance (i.e., CPU time to produce the stiffness) has not yet been evaluated directly in a finite element application, nor has it been evaluated against the typical linear elastic response used in a conventional finite element model. Those metrics are also important, however, the method described provides a valuable proof of concept. There is still progress to be made in improving

the accuracy of the machine learning model and also progress is desired especially with regard to prediction of non-linear constitutive relationships that would typically be of interest in a multi-scale finite element model. The future of this work aims to include composite allowables or strength prediction along with the initial stiffness computation.

Acknowledgments

Thanks to NASA's Space Launch System (SLS) for funding this work.

References

[1] <https://gitlab.grc.nasa.gov/joshua.stuckner/pyply>

[2] Murthy, P. L. N., Pineda, E. J. (2016). Tool for generation of MAC/GMC representative unit cell for CMC/PMC analysis. *NASA/TM—2016-219127*.

[3] <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

[4] Aboudi, J. et al. (2013) *Micromechanics of Composite Materials*. Waltham, MA: Elsevier. ISBN-13: 978-0-12-397035-0

[5] Daniel, I. M. (2006) *Engineering Mechanics of Composite Materials*. Oxford, NY: Oxford University Press. ISBN 978-0-19-515097-1

[6] Brownlee, J. (2019) *Deep Learning with Python*. E-book.