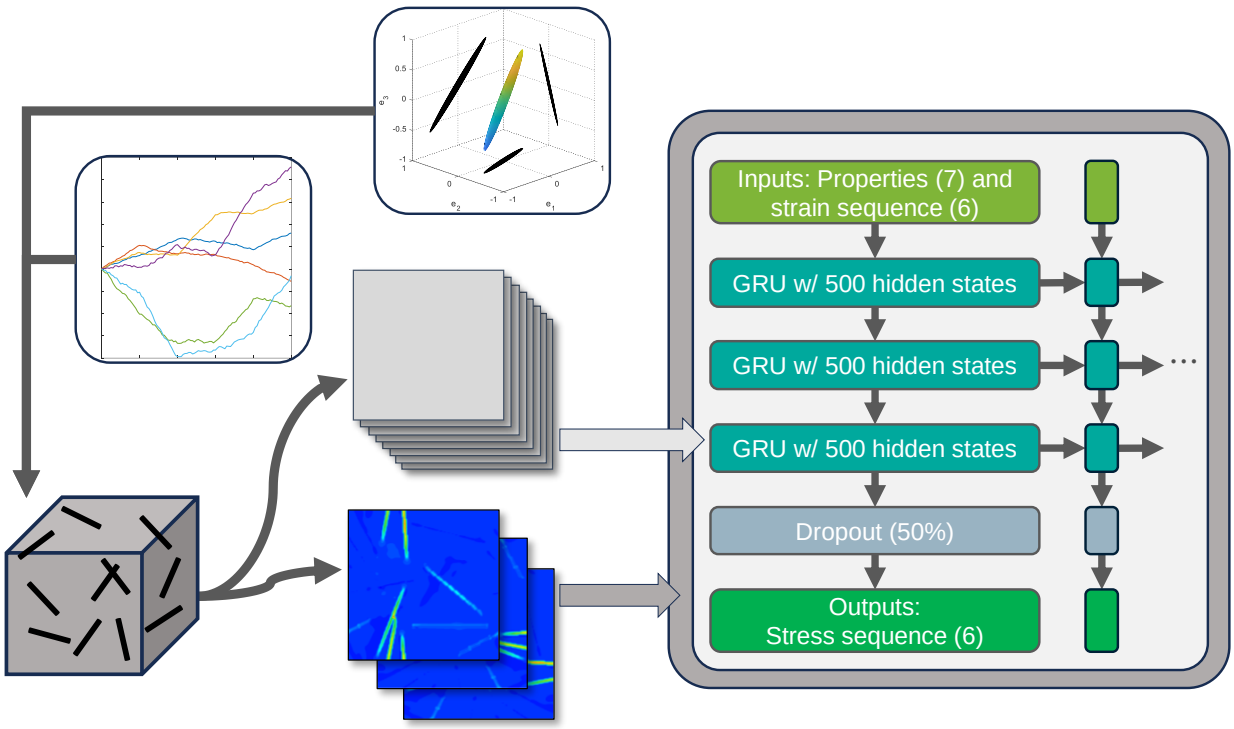
CHALMERS
UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG



Micromechanics-based Artificial Neural Networks and Transfer Learning for Modeling Short Fibre Reinforced Composites in Automotive Applications

Master's thesis in Mobility Engineering

HON LAM CHEUNG

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Micromechanics-based Artificial Neural Networks and Transfer Learning for Modeling Short Fibre Reinforced Composites in Automotive Applications

Enhancing pre-trained artificial neural networks performance for short fibre reinforced composite elasto-plastic response prediction using data collected from RVE simulations

Hon Lam CHEUNG

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Mechanics of Materials
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2023

Micromechanics-based Artificial Neural Networks and Transfer Learning for Modeling Short Fibre Reinforced Composites in Automotive Applications
HON LAM CHEUNG

Supervisor and Examiner:
Mohsen Mirkhalaf, Department of Physics, Gothenburg University

Master's Thesis 2023
Department of Physics
Mechanics of Materials
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Visualisation of the data generation and transfer learning approach for enhancing the micromechanical-based artificial neural network.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Micromechanics-based Artificial Neural Networks and Transfer Learning for Modeling Short Fibre Reinforced Composites in Automotive Application

HON LAM CHEUNG
Department of Physics
Chalmers University of Technology

Abstract

For the automotive industry, improving efficiency is crucial while weight reduction is one key factor for high efficiency. Short Fiber Reinforced Composites (SFRCs) provide superior performance at reduced weight and are suitable for mass production, making them attractive materials for the industry. However, the mechanical modeling of SFRCs poses challenges. A full field analysis may take multiple trials to generate a proper realization, and subsequent analysis can take hours or days to finish. Furthermore, the mechanical response is influenced by fiber orientation and volume fraction, which can have countless configurations. Therefore, data-driven models for SFRCs have gained popularity. Previous work utilized mean-field analysis results to train a recurrent neural network, aiming to predict elasto-plastic stress response of SFRCs with different strain paths and properties. This study enhanced the mean-field network with a limited amount of full-field data, aiming to improve the network's prediction accuracy to a full-field level.

Keywords: Transfer learning, Recurrent neural networks, Short fiber reinforced composite, Full field simulation, High fidelity.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to Professor Mohsen Mirkhalaf. He is always supportive and provides invaluable feedback on my work. His attention to detail also motivated me to perform at a higher standard.

I would also like to extend my appreciation to all those who have contributed, directly or indirectly, to the successful completion of this thesis. This is a memorable journey because of you.

Lastly, I would like to thank my parents for their kind support. This journey would not be possible without them.

Hon Lam Cheung, Gothenburg, September 2023

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

ADAM	Adaptive Moment Estimation
ANN	Artificial Neural Network
BRNN	Bayesian Recurrent Neural Network
FE	Finite Element
FFNN	Feed-Forward Neural Network
FFT	Fast-Fourier-Transforms
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
MaRE	Maximum Relative Error
MeRE	Mean Relative Error
MFH	Mean-Field Homogenization
ODF	Orientation Distribution Function
RNN	Recurrent Neural Network
RVE	Representative Volume Element
SFRC	Short fiber reinforced composite
SGD	Stochastic Gradient Descent
UD	Uni-Directional

Contents

List of Figures

List of Tables

1
Introduction

Efficiency has become crucial in the mobility industry due to emissions regulations and rising energy costs. Besides electrification, weight reduction plays a significant role in improving efficiency [3]. Composite materials provide an effective solution for weight reduction [4] and have been extensively employed in the aerospace industry, with increasing popularity in the automotive industry [5]. Short Fiber Reinforced Composites (SFRCs) are attractive for several reasons: (i) they can be recycled, reducing material costs [6]; (ii) they are suitable for low-cost and efficient manufacturing processes like injection molding and hot pressing [7]; and (iii) they can be used to produce complex parts. SFRCs find applications in various components, including interior and certain structural parts in the mobility industry [4]. Given that parts made from SFRCs are generally produced by injection moulding or extrusion, the fiber orientation depends heavily on material flow during manufacturing. This orientation significantly influences the mechanical responses of SFRC parts and needs to be considered in the modeling. While one can directly model an SFRC part with the microscopic details, this overly complex approach would require an enormous amount of computational resources and time to perform simulation for large structures. When one aims to model the full structure while also considering the micro-structure properties, multi-scale models are used. These models couple the macro-model (the structure/ part of interest) with the micro-model (the microscopic material model). While multi-scale models do improve efficiency, they remain computationally expensive [8].

Homogenization schemes such as mean-field and full-field analysis serve as the micro-model to calculate SFRCs' micro-mechanical responses. Although mean-field analysis is fast, it has lower fidelity since it approximates the material's average response based on some assumptions [9]. On the other hand, full-field analysis requires the generation of the actual micro-structure and utilizes Finite Element (FE) or Fast-Fourier-Transforms (FFT) analysis for computation. Full-field analysis has a higher fidelity as it computes detailed micro-fields. However, generating a micro-structure with desired properties can be challenging, and the analysis can take hours or even days to solve depending on the complexity. Therefore, in applications that require continuous evaluation of the micro-models (e.g., models evaluating the non-linear material behaviour such as elasto-plastic response), multi-scale modeling, especially with full-field analysis, is still too computationally expensive to employ. The major impediment for multi-scale models is the computational cost related to evaluating the micro-model at each integration point of the macro-model [10].

Data-driven approaches could help reduce the time required for the evaluation of the micro-models while keeping the accuracy of full-field analysis. Recently, Artificial Neural Networks (ANNs) have gained a strong interest in engineering fields, including utilizing them to accurately predict composite materials' properties (see review [11, 12]). However, since ANNs simply map inputs to desired outputs based on their training data, a large amount of data is required to properly train them. For example, Friemenn et al. [7, 1] used a data set with 40,000 data to train an ANN with a similar purpose, making it unrealistic to gather sufficient full-field data to train an ANN from scratch. Transfer learning could be a potential solution to achieve a full-field level of prediction accuracy for SFRCs' elasto-plastic response using ANNs. This approach involves transferring and reusing the knowledge learned from an existing ANN for a similar aim. Transferring this knowledge to a new ANN can reduce the need for full-field training data [13].

1.1 Previous Related Studies

Efforts have been made to use data-driven methods, specifically ANNs, to accelerate the modeling of SFRCs. Mentges et al. [14] developed an ANN to predict the stiffness tensor of SFRCs within the linear elastic range. They employed a two-step approach that combined uni-directional Representative Volume Element (RVE) and orientation averaging to generate data, reducing the computational cost. Similarly, Breuer and Stommel [9] developed a similar ANN, but with data directly generated from RVE simulations. While both studies demonstrated the potential of using ANNs in SFRC modeling, they considered only the elastic domain. In the elaso-plastic domain, Wu et al. [15] developed a Recurrent Neural Network (RNN) with Gated Recurrent Unit (GRU) to serve as a surrogate model of micro-models. They collected data from 2D RVEs of continuous fiber reinforced elasto-plastic matrix composite subjected to random strain path loading. Ghavamian et al. [10] also developed an RNN for a similar aim, but they sampled only a small subspace of all possible strain sequences to reduce the computational cost. Notably, both studies by Wu and Ghavamian considered a single fiber volume fraction and relied on two-dimensional RVEs. On the other hand, Mozaffar et al. [16] considered the composites' properties in the RNN inputs to predict material plasticity. However, they too were limited to two-dimensional RVEs. Collectively, these studies showed the feasibility of predicting the path-dependent eleasto-plastic response of SFRCs using RNNs. Addressing the computationally challenges posed by 3D RVE, Jung et al. [17] employed a transfer learning approach to enhance ANNs' predictions of elasto-plastic response in SFRCs. They showed transfer learning can improve the ANNs' performance even with limited full-field data. However, they exclusively focused on SFRCs with uni-directional inclusion orientation and uni-axial loading conditions. Given that a single loading condition was considered for each neural network, a feed-forward neural network was considered sufficient. Therefore, the developed networks can only predict stress response for the loading conditions they were trained on.

This study is based on the work conducted by Friemenn et al. [7, 1]. They developed

a deep RNN with 3 hidden GRU layers to predict the stress sequences. The RNN inputs consisted of the fiber orientation tensor, volume fraction, and strain sequences of an SFRC. They utilized mean-field homogenization to generate a data set for the network to train on. Each data has a randomly sampled fiber orientation tensor, volume fraction and random strain path. The developed RNN can accurately predict the stress obtained from mean-field analysis. This RNN serves as the starting point of the transfer learning approach in this study.

1.2 Contribution

This study introduces the use of transfer learning to develop an ANN capable of predicting the elasto-plastic response of SFRCs at full-field accuracy. By employing the transfer learning approach, a small amount of full-field data, in comparison to the amount of mean-field data used to develop the original network, is sufficient. This approach significantly reduces the computational cost of generating training data. The results demonstrate that the final ANN is capable of predicting SFRCs' elasto-plastic response with a low relative error compared to results obtained from the full-field analysis. Moreover, a substantial prediction performance improvement is observed in the final ANN compared to the network developed with a small amount of full-field data.

1.3 Outline

The remainder of this thesis is organized as follows. Chapter 2 introduces the theory of micro-mechanics modeling and ANNs. Chapter 3 explains the data generation process for the high-fidelity full-field data set. Chapter 4 presents the development process of the RNN, the transfer learning approach and evaluation methods. Chapter 5 demonstrates the results obtained from the developed RNN and compares different networks. Chapter 6 provides related discussions and concludes this study.

2

Theory

This chapter introduces the theories needed for modeling SFRCs, followed by an overview of artificial neural networks, specifically the feed-forward neural networks and recurrent neural networks, and their training methods.

2.1 Micro-mechanics Modeling for Short Fibre Reinforced Composite

In this section, the tools for modeling SFRCs are introduced. Firstly, the orientation tensor, which describes SFRCs properties, is introduced. Following that, the mean-field and full-field simulations for predicting SFRCs' response are discussed.

2.1.1 Orientation Tensor

The orientation of the fibres inside each small volume (i.e, an RVE) can vary widely. A tool is needed to describe fiber orientations. Consider a single cylindrical fiber. Its orientation can be described by two angles, θ and ϕ, as illustrated in Figure 2.1. And a unit vector $\boldsymbol{p}$ can be constructed as

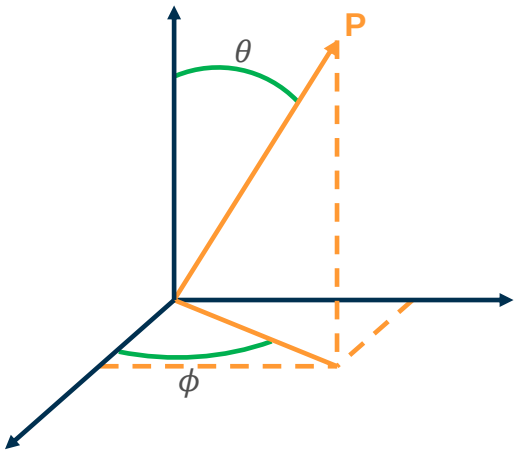


Figure 2.1: Unit vector $\boldsymbol{p}$ defined by θ and ϕ.

$$\boldsymbol{p} = [sin(\theta)cos(\phi)\ sin(\theta)sin(\phi)\ cos(\theta)]^T. \tag{2.1}$$

Inside an RVE, each inclusion could have a different unit vector $\boldsymbol{p}$. Therefore, the Orientation Distribution Function (ODF) $\psi(\boldsymbol{p})$ is introduced. This function

represents the likelihood of finding an inclusion in the interval from $\boldsymbol{p}$ to $\boldsymbol{p} + d\boldsymbol{p}$. Every ODF satisfies the following conditions:

$$\psi(\boldsymbol{p}) = \psi(-\boldsymbol{p}), \tag{2.2}$$

since inclusions defined by unit vector $\boldsymbol{p}$ and $-\boldsymbol{p}$ are identical. Additionally,

$$\oint \psi(\boldsymbol{p}) d\boldsymbol{p} = 1, \tag{2.3}$$

serve as a normalization condition to ensure the sum of probabilities equals 1.

However, ODF is not necessary in some use cases, and an even-order orientation tensor is used. In this study, a second-order orientation tensor ($\boldsymbol{a}$) is used by Digimat-FE [18] to generate RVE, which can be defined as

$$a_{ij} = \oint p_i p_j \psi(\boldsymbol{p}) d\boldsymbol{p}, \tag{2.4}$$

or it can be expressed as the orientation averages as

$$\boldsymbol{a} = \langle \boldsymbol{p} \otimes \boldsymbol{p} \rangle , \tag{2.5}$$

where $\langle\rangle$ denotes an averaging process over all fibers within the RVE. Since for each inclusion $p_i p_j$ equals $p_j p_i$, the orientation tensor is symmetrical ($a_{ij} = a_{ji}$). Moreover, due to the normalization condition for ODF, the trace of the orientation tensor is equal to one. More detailed information about orientation tensors can be found in [19].

2.1.2 Mean-field Homogenization

Mean field homogenization aims to describe the relationship between the macro stress $\boldsymbol{\sigma}_M$ and macro strain $\boldsymbol{\varepsilon}_M$ of a material, i.e.

$$\boldsymbol{\sigma}_M = \mathbb{C}_M : \boldsymbol{\varepsilon}_M. \tag{2.6}$$

Considering a two-phase composite material, with uniform constitutive material stiffness tensors $\mathbb{C}_0$ and $\mathbb{C}_1$ for the matrix and inclusions, respectively. The macro strain and stress tensor can be written in terms of the volume-averaged strain $\langle \boldsymbol{\varepsilon}_m \rangle$ and volume-averaged stress $\langle \boldsymbol{\sigma}_m \rangle$ over the matrix subdomain ω_0 and inclusions subdomain ω_1 as [20]

$$\boldsymbol{\varepsilon}_M = v_0 \langle \boldsymbol{\varepsilon}_m \rangle_{\omega_0} + v_1 \langle \boldsymbol{\varepsilon}_m \rangle_{\omega_1} \tag{2.7}$$

and

$$\boldsymbol{\sigma}_M = v_0 \langle \boldsymbol{\sigma}_m \rangle_{\omega_0} + v_1 \langle \boldsymbol{\sigma}_m \rangle_{\omega_1} . \tag{2.8}$$

where v_0 and v_1 are volume fractions for the matrix and fiber phase, respectively, and $v_0 + v_1 = 1$. In addition, a strain concentration tensor $\mathbb{B}^{\varepsilon}$ describe the relation between the strain averages per phase as

$$\langle \boldsymbol{\varepsilon}_m \rangle_{\omega_1} = \mathbb{B}^{\varepsilon} : \langle \boldsymbol{\varepsilon}_m \rangle_{\omega_0} \tag{2.9}$$

For the Mori-Tanaka model [21], the strain concentration tensor is approximated as

$$\hat{\mathbb{B}}^{\varepsilon} = \left\{ \mathbb{I} + \mathbb{S} : [(\mathbb{C}_0)^{-1} : \mathbb{C}_1 - \mathbb{I}] \right\}^{-1}, \tag{2.10}$$

where $\mathbb{I}$ is the fourth order identity tensor, and $\mathbb{S}$ is the Eshelby tensor [22] which depends on the inclusion geometry and $\mathbb{C}_0$. Using Equations (2.7)-(2.10), the resulting homogenized macroscale stiffness tensor can be written as

$$\hat{\mathbb{C}}_M = [v_1 \mathbb{C}_1 : \hat{\mathbb{B}}^{\varepsilon} + v_0 \mathbb{C}_0] : [v_1 \hat{\mathbb{B}}^{\varepsilon} + v_0 \mathbb{I}]^{-1} \tag{2.11}$$

2.1.3 Computational Homogenization

Full-field simulation generates an RVE that contains all the necessary information about the micro-structure. Appropriate boundary conditions and desired loadings are introduced. Then, using numerical methods (e.g., FE or FFT methods) to discretize the domain and compute the micro-mechanical response. Since full-field simulation can include more micro-structure information and uses fewer assumptions, it can obtain more accurate predictions compared to mean-field homogenization [9]. The first step of the full-field simulation is to generate the RVE geometry by randomly placing the inclusion (see e.g., [23]). Additionally, periodic models can achieve better prediction [24], in such cases, periodic geometry is needed. Figure 2.2 displays an example of SFRC RVE geometry generated by Digimat-FE. Then FE

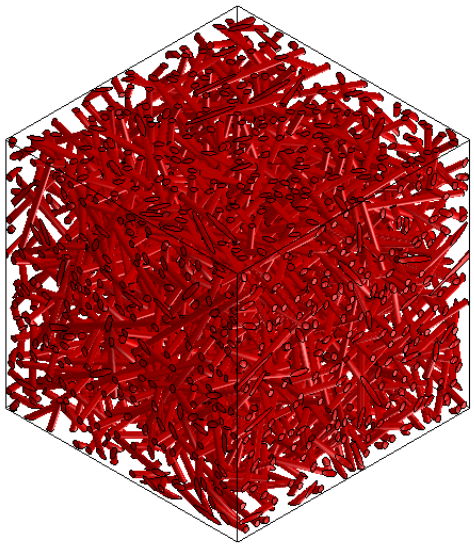


Figure 2.2: Example of an SFRC RVE realization.

or FFT analysis can be utilized to compute the behaviour under prescribed loading conditions. In FE analysis, the RVE geometry is discretized into simple idealized elements (e.g., tetrahedral element) and shape functions are used to approximate the stiffness matrix, stress and strain within the element. This enables the computer to calculate their behaviour under loading. Periodic boundary conditions can be utilized for better predictions. See [25] for more information on the theory of FE

analysis and [26] for more information on how boundary conditions and loading can be applied to FE micro-mechanical simulations.

FFT analysis was first proposed by Moulinec and Suquet [27] for micro-mechanical modeling as an alternative method to FE analysis. It is efficient since the kernel is applied in the Fourier domain by optimized FFT routines, meshing is avoided, and the system/stiffness matrix does not have to be assembled. See [28, 29] for more information on the theory of the FFT analysis. More information about computational homogenization can be found in [8, 30].

2.2 Artificial Neural Networks

In this section, an overview of artificial neural networks is provided. Firstly, artificial neurons, feed-forward neural networks, and training methods are introduced. Then, recurrent neural networks and the challenges faced during training are presented, followed by an introduction to the state-of-the-art gated recurrent unit. Finally, various optimizers for training are discussed. More information about the theory of artificial neural networks can be found in [31].

2.2.1 Artificial Neurons

Artificial neural networks emulate the behaviour of neurons with artificial neurons, where the input of artificial neurons is connected to other neurons or external input. Each connection is assigned a weight, representing the strength of the correlation between input and output [32]. Figure 2.3 illustrates the schematic of a single artificial neuron. Artificial neurons calculate the weighted sum of the inputs with a

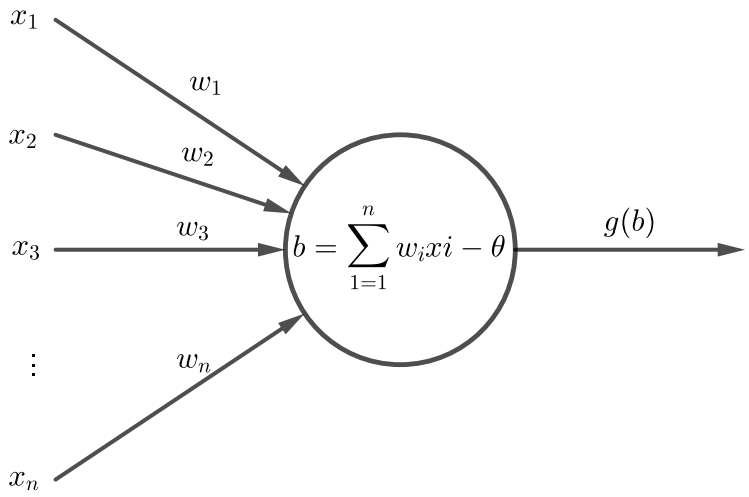


Figure 2.3: Schematic of a single artificial neuron.

bias subtracted from it, which is referred to as the local field of a neuron and can be expressed as

$$b = \sum_i w_i x_i - \theta, \tag{2.12}$$

where b represents the local field, w_i denotes the weight associated with input x_i, i is index for of number of inputs, and θ represents the bias value. Finally, an activation function is applied to process the local field. The activation function can be linear or non-linear. However, non-linear activation is needed to solve non-linear problems [33]. Moreover, since back propagation is often used to train the artificial neural network, the activation function has to be smooth and continuous in most cases. The output of the neuron O can be expressed as

$$O = g(b), \tag{2.13}$$

where $g(b)$ is the activation function processing local field b.

2.2.2 Feed Forward Neural Networks

Typically, a Feed-Forward Neural Network (FFNN) is composed of multiple layers of artificial neurons. The neurons in one layer are connected to both the layer before and the layer after. A representation of a feed-forward neural network is shown in Figure 2.4. Layers between input and output layers are called hidden

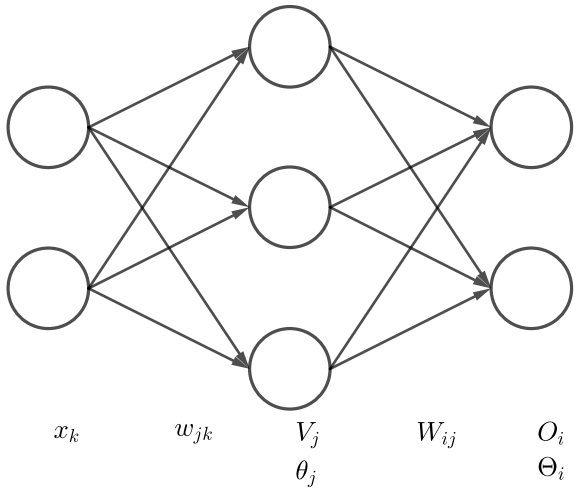


Figure 2.4: Example of a fully connected feed-forward neural network. This FFNN has an input layer, one hidden layer, and an output layer.

layers. According to the universal approximation theorem, one hidden layer is sufficient to represent a continuous function [34]. However, for more complex problems, additional hidden layers may be required or beneficial in order to increase the efficiency of the neural network. Neural networks with more than two hidden layers are commonly referred to as deep neural networks.

2.2.3 Gradient Descent and Back Propagation

Supervised learning, where users provide data with both inputs and targets, is one of the most commonly used approaches to train a neural network. In supervised learning, gradient descent and back propagation are commonly employed to minimize the loss between the target and output by adjusting the trainable parameters

(i.e., weights and bias) in each neuron. Using the network structure shown in Figure 2.4 as an example, the weight adjustment process can be expressed as [31]

$$\delta W_{mn} = -\eta \frac{\partial L}{\partial W_{mn}}, \tag{2.14}$$

where W represents the weight in the last layer, the subscript m and n donate the number of neurons in that layer and the connection from the previous layer the weight is associated with, respectively. L is the loss function used to monitor the performance of the network by comparing the target and output, which has a minimum if the desired output is reached, and η is the learning rate. By applying the chain rule, the weight update equation can be derived as

$$\delta W_{mn}^{(\mu)} = \eta (t_m^{(\mu)} - O_m^{(\mu)}) g'(B_m^{(\mu)}) V_n^{(\mu)}, \tag{2.15}$$

where t, O, and B are the target, output and local field, respectively. The subscript m represents the number of neurons, and superscript μ represents the number of data. While V is the output from the previous layer, subscript m represents the number of neurons in the previous layer. The weighted output errors $\Delta_m^{(\mu)}$ is defined as

$$\Delta_m^{(\mu)} = (t_m^{(\mu)} - O_m^{(\mu)}) g'(B_m^{(\mu)}). \tag{2.16}$$

Then, Equation (2.15) can be rewritten as

$$\delta W_{mn}^{(\mu)} = \eta \Delta_m^{(\mu)} V_n^{(\mu)}. \tag{2.17}$$

For weights in the previous layer, w, the weight adjustment can be expressed as

$$\delta w_{mn} = \eta \sum_{\mu i} (t_i^{(\mu)} - O_i^{(\mu)}) \frac{\partial O_i^{(\mu)}}{\partial w_{mn}}, \tag{2.18}$$

$$\delta w_{mn} = \eta \sum_{\mu i} (t_i^{(\mu)} - O_i^{(\mu)}) \sum_l \frac{\partial O_i^{(\mu)}}{\partial V_l^{(\mu)}} \frac{\partial V_l^{(\mu)}}{\partial w_{mn}}, \tag{2.19}$$

$$\delta w_{mn}^{(\mu)} = \eta \sum_i (t_i^{(\mu)} - O_i^{(\mu)}) g'(B_i^{(\mu)}) W_{im} g'(b_m^{(\mu)}) x_n^{(\mu)}. \tag{2.20}$$

With the weighted errors $\delta_m^{(\mu)}$ is defined as

$$\delta_m^{(\mu)} = \sum_i (t_i^{(\mu)} - O_i^{(\mu)}) g'(B_i^{(\mu)}) W_{im} g'(b_m^{(\mu)}), \tag{2.21}$$

$$\delta_m^{(\mu)} = \sum_i \Delta_i^{(\mu)} W_{im} g'(b_m^{(\mu)}). \tag{2.22}$$

Then, equation (2.20) can be rewritten as

$$\delta w_{mn}^{(\mu)} = \eta \delta_m^{(\mu)} x_n^{(\mu)}. \tag{2.23}$$

Note that in the Equation (2.22), the weighted error $\delta_m^{(\mu)}$ incorporates the weighted output error $\Delta_m^{(\mu)}$ from the next layer. Therefore, the computation of error in neural

networks begins from the output layer and propagates backwards to the previous layer. This method to compute gradient is commonly referred to as back propagation. The same approach is used to adjust the bias to minimize the loss value. Since the bias is subtracted from the weighted sum in (2.12) and there is no multiplication involved, the bias update equations can be expressed as

$$\delta\Theta_m^{(\mu)} = -\eta\Delta_m^{(\mu)} \quad \text{and} \quad \delta\theta_m^{(\mu)} = -\eta\delta_m^{(\mu)}. \tag{2.24}$$

When using back propagation to compute the gradients $\frac{\partial H}{\partial w}$ and $\frac{\partial H}{\partial \theta}$, issues such as vanishing and exploding gradients can arise, particularly in deep neural networks with three or more hidden layers, as shown in Figure 2.5. If the bias of each neuron

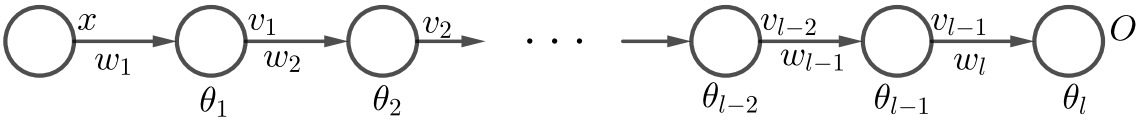


Figure 2.5: Example of a deep neural network with one neuron in each layer.

is assumed to be 0 and all neurons have the same activation function $g(x)$. The output O with respect to input x can be expressed as

$$O = g(w_l \cdot g(w_{l-1} \cdot g(w_{l-2} \cdots g(w_2 \cdot g(w_1 x))))). \tag{2.25}$$

Assume linear activation function $g(x) = x$ is used, then the output simplifies to

$$O = w_l w_{l-1} w_{l-2} \cdots w_2 w_1 x. \tag{2.26}$$

The gradient of output O with respect to w_n can be expressed as [31]

$$\frac{\partial O}{\partial w_n} = \left(\prod_{i=1, i\neq n}^{l} w_i \right) x. \tag{2.27}$$

If the weights w_i are greater than 1, the gradient $\frac{\partial O}{\partial w_n}$ increases exponentially with the number of layers. In contrast, if the weights w_i are less than 1, the gradient $\frac{\partial O}{\partial w_n}$ decreases exponentially. This phenomenon can make it challenging to update the parameters with gradient descent. Common methods to combat this problem include weight initialization using Xavier Glorot's initialization [35], normalization (such as layer normalization) [36], gradient clipping etc.

2.2.4 Recurrent Neural Networks

A feed-forward neural network is typically designed to map one input to one output. FFNN can perform time series prediction using a sliding window technique (see e.g., [37]), meaning a fixed number of previous series values are used to predict the next value. However, if the sequence is longer than the number of inputs, values from earlier time steps need to be removed from the inputs. Therefore, this technique is not suitable for all problems. For example, the path-dependent elasto-plastic response since the whole strain sequences are necessary for the prediction.

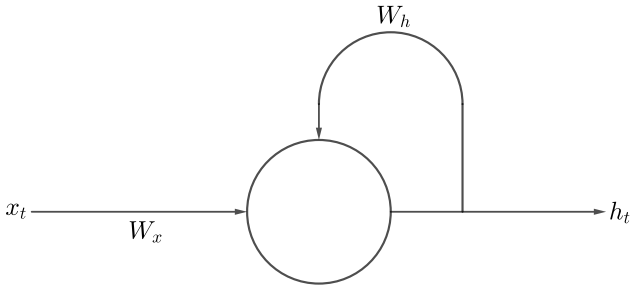


Figure 2.6: A recurrent neuron that takes the previous output as an input.

To address this limitation, a solution would be to connect the output of a neuron to its input to introduce feedback, creating an RNN architecture as illustrated in Figure 2.6. The recurrent neural network can memorize the previous inputs via the feedback connection, and utilize both current and previous inputs to determine the current state. Therefore, the network can be trained to make predictions for sequential data.

2.2.4.1 Back Propagation Through Time

The training of an RNN can be performed using back propagation. Consider a many-to-many RNN where each input corresponds to one output as an example, Figure 2.7 illustrates an unfolded many-to-many RNN over time. The total loss can

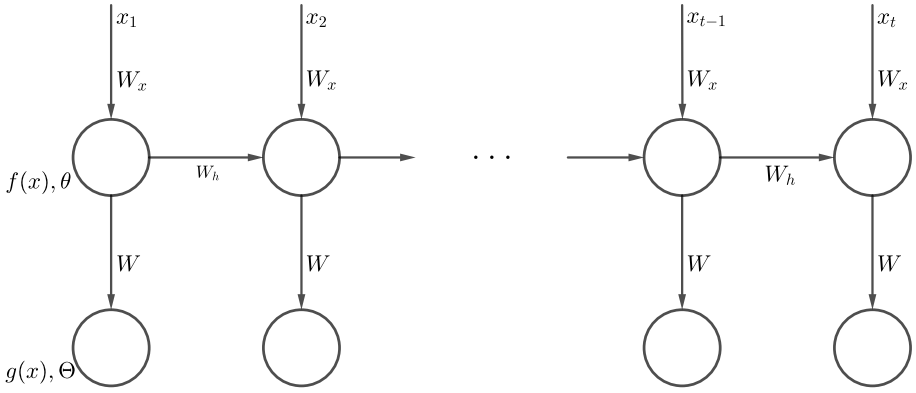


Figure 2.7: Recurrent neuron after unfolding through time.

be calculated as the sum of individual losses as

$$L = \frac{1}{T}\sum_{t=1}^{T} l_t, \tag{2.28}$$

where L is the total loss, l donates the individual loss, subscript t represents the time step in the sequence, and T represents the total number of time step. The hidden state of the recurrent neuron and the output of the output neuron can be calculated as

$$h_t = f(W_x x_t + W_h h_{t-1} - \theta) \tag{2.29}$$

and

$$O_t = g(W h_t - \Theta), \tag{2.30}$$

where h, and θ are the hidden state, and bias for the recurrent neuron, respectively. O is the output of the output neuron, and subscript t represents the time step. The parameters W_x, and W_h are the weight associated with the input and previous hidden state, respectively. W is the weight of the output neuron. Finally, f and g are the activation functions of recurrent and output neurons, respectively. Using b_t and B_t to represent the local field of recurrent and output neurons, respectively. Equation (2.29) and (2.30) can be rewritten as

$$h_t = f(b_t) \tag{2.31}$$

and

$$O_t = g(B_t). \tag{2.32}$$

To adjust w_h, the gradient $\frac{\partial L}{\partial w_h}$ is needed, and it can be expressed as

$$\frac{\partial L}{\partial w_h} = \frac{1}{T}\sum_{t=1}^{T} \frac{\partial l_t}{\partial O_t}\frac{\partial g(B_t)}{\partial h_t}\frac{\partial h_t}{\partial w_h}. \tag{2.33}$$

And $\frac{\partial h_t}{\partial w_h}$ can be expressed as [38]

$$\frac{\partial h_t}{\partial w_h} = \frac{\partial f(b_t)}{\partial w_h} + \sum_{i=1}^{t-1}\left(\prod_{j=1+1}^{t}\frac{\partial f(b_j)}{\partial h_{j-1}}\right)\frac{\partial f(b_i)}{\partial w_h}. \tag{2.34}$$

Although back propagation can be applied to RNN, Equation (2.34) has demonstrated that the recursive multiplication for a long time series resembles that of a deep neural network. Consequently, RNN is prone to the vanishing or exploding gradient problem. Additionally, this implies that in a long sequence, the earlier time steps are challenging to train, resulting in the short-term memory problem of an RNN.

2.2.4.2 Gated Recurrent Unit

Units with gating mechanisms are commonly used to avoid the vanishing gradient problem. Long Short-Term Memory (LSTM) is designed to use multiplicative gates, such that information can be stored and accessed by memory cells over a long sequence [39]. Gated Recurrent Unit (GRU) was developed in more recent years, it was motivated by LSTM but with a simpler to compute and implement hidden unit [40]. Figure 2.8 has shown the schematic of a GRU, which includes a reset gate and an update gate. In a GRU, the reset gate $\boldsymbol{r}_t$ is calculated using the input $\boldsymbol{x}_t$ and the previous hidden $\boldsymbol{h}_{t-1}$ as

$$\boldsymbol{r}_t = \sigma\left(\boldsymbol{W}_r\boldsymbol{x}_t + \boldsymbol{b}_r + \boldsymbol{R}_r\boldsymbol{h}_{t-1}\right), \tag{2.35}$$

where $\boldsymbol{W}_r$, $\boldsymbol{R}_r$, and $\boldsymbol{b}_r$ represent the input weights, recurrent weights, and biases for the reset gate. Then, the reset gate $\boldsymbol{r}_t$, input $\boldsymbol{x}_t$, and previous hidden state $\boldsymbol{h}_{t-1}$ are utilized to compute the candidate hidden state $\widetilde{\boldsymbol{h}}_t$ as

$$\widetilde{\boldsymbol{h}}_t = \tanh\left(\boldsymbol{W}_{\widetilde{h}}\boldsymbol{x}_t + \boldsymbol{b}_{\widetilde{h}} + \boldsymbol{r}_t \odot \left(\boldsymbol{R}_{\widetilde{h}}\boldsymbol{h}_{t-1}\right)\right), \tag{2.36}$$

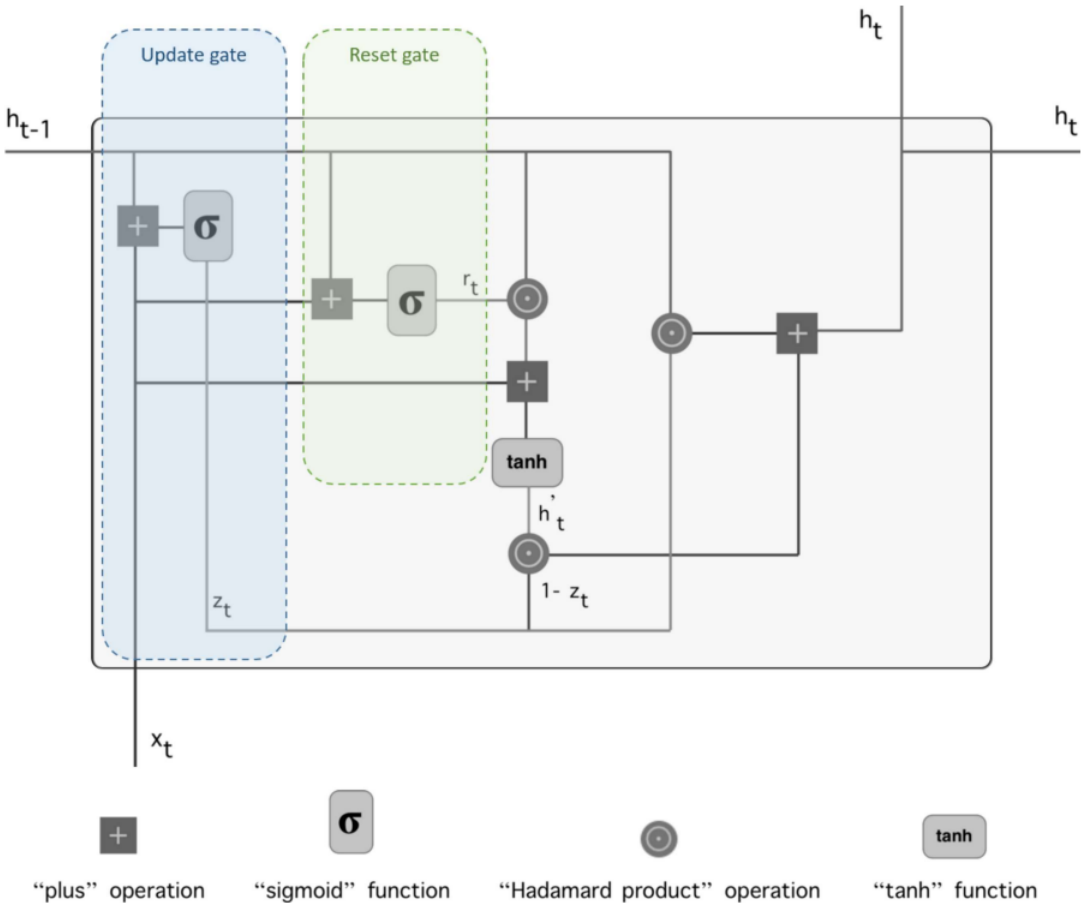


Figure 2.8: Schematic of a GRU (Figure taken from [1]).

where $\boldsymbol{W}_{\tilde{h}}$, $\boldsymbol{R}_{\tilde{h}}$, and $\boldsymbol{b}_{\tilde{h}}$ donate the input weight, recurrent weight and bias for candidate hidden state, respectively. Furthermore, the update gate $\boldsymbol{z}_t$ is calculated as

$$\boldsymbol{z}_t = \sigma\left(\boldsymbol{W}_z \boldsymbol{x}_t + \boldsymbol{b}_z + \boldsymbol{R}_z \boldsymbol{h}_{t-1}\right), \tag{2.37}$$

where $\boldsymbol{W}_z$, $\boldsymbol{R}_z$, and $\boldsymbol{b}_z$ represent the input weight, recurrent weight, and bias for the update gate, respectively. Finally, the hidden state $\boldsymbol{h}_t$ is calculated as

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) \odot \widetilde{\boldsymbol{h}}_t + \boldsymbol{z}_t \odot \boldsymbol{h}_{t-1}. \tag{2.38}$$

The candidate hidden state is forced to ignore the previous hidden state if the reset gate is 0. This mechanism enables the unit to forget past information that is irrelevant to the future. Additionally, to assist long-term memory within the unit, the update gate controls the amount of information preserved from the previous hidden state. Figure 2.9 illustrates how reset gate and update gate control the flow of information within a GRU.

2.2.5 Optimizers

An optimizer is an algorithm used to adjust the trainable parameters in a neural network, for example, gradient descent [41]. However, standard gradient descent

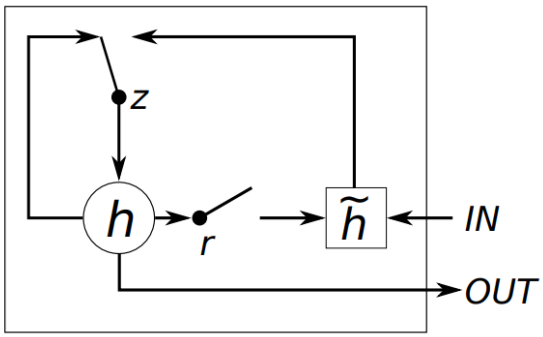


Figure 2.9: Flow of information inside GRU gated by update and reset gate (Figure reproduced from [2]).

updates the parameters only after processing the entire data set, commonly referred to as one epoch, making it inefficient when dealing with a large amount of data. Stochastic Gradient Descent (SGD) is a more efficient variant of gradient descent [42]. SGD algorithm randomly selects a subset of data, known as a mini-batch, and then performs gradient descent to update the parameters, resulting in a slightly improved iteration of the neural network. By updating the neural network more frequently with mini-batches, stochastic gradient descent can converge towards the minimum loss more efficiently.

Adaptive Moment Estimation (ADAM) [43] is another widely used optimizer. It operates by maintaining the element-wise moving average of the gradient term, and the squared gradient. The equations for these moving averages are as

$$m_l = \beta_1 m_{l-1} + (1 - \beta_1)[\delta L/\delta\theta_l] \tag{2.39}$$

and

$$v_l = \beta_2 v_{l-1} + (1 - \beta_2)[\delta L/\delta\theta_l]^2, \tag{2.40}$$

where m is the moving average of the gradient term, v is the moving average of the squared gradient term, θ is the parameter to be adjusted, subscript l represents the iteration, L represent the loss function, and β_1 and β_2 are the decay rates for the moving averages. Then, the parameters are updated using the following equation:

$$\theta_{l+1} = \theta - \frac{\alpha m_l}{\sqrt{v_l} + \epsilon}, \tag{2.41}$$

where α represents the learning rate, and ϵ is a small value added to avoid division by zero. Adding the momentum-like term m_l in ADAM is a technique to increase efficiency [42]. It achieves this by incorporating a fraction β_1 of the previous momentum to the current update vector. Mimicking the effect of an object rolling down the slope, the object gains speed and momentum along that direction, enabling the object to reach the bottom of a slope faster. The parameter β_1 represents the resistance that should be smaller than 1 to ensure convergence of the terminal speed is converging. Typically, β_1 is set to 0.9. On the other hand, the moving average of the squared gradient (i.e., root-mean-square) term reduces the effective learning rate

for large gradients and increases it for small gradients [42]. This helps dampen the oscillation of the update vector, allowing for a higher learning rate and more efficient learning. The squared gradient decay factor, β_2, is usually set to 0.99. Additionally, to avoid division by zero when v_l is small, a small value ϵ (typically set at $1e-8$) is added in the denominator in Equation (2.41).

3
Data Generation

In this study, an elasto-plastic matrix and an elastic fiber are modeled, and a variety of fiber orientations and fiber volume fractions are considered. Constitutive properties of fiber and matrix (see [44, 1]) are listed in Table 3.1. As supervised learning is

Table 3.1: Constitutive properties of elasto-plastic matrix and elastic fibers.

Fiber properties		**Matrix properties**	
Young's modulus	76 GPa	Young's modulus	3.1 GPa
Poisson's ratio	0.22	Poisson's ratio	0.35
Length	240 μm	Yield strength	25 MPa
Diameter	10 μm	Linear hardening modulus	150 MPa
Fiber volume fraction	0.1-0.15	Hardening modulus	20 MPa
		Hardening exponent	325

employed to train the neural network, a full-field data set that covers a wide range of fiber orientations and loading conditions is required. A data generation process is followed to generate this data. Initially, a random orientation tensor, fiber volume fraction, and 6D random strain path are sampled. Then, Digimat-FE is used to generate the RVE geometry and solve the boundary value problem using either an FE or FFT analysis. The subsequent subsections provide a detailed explanation of each step involved in this process.

3.1 Random Orientation Tensor Generation

This study utilizes a procedure for generating random reference orientation tensor employed by Friemann et al. [7, 1]. For a 3D orientation tensor, first, a set of three positive numbers that sum up to one is sampled, representing the principal components of the orientation tensor. These three numbers constitute a diagonal orientation tensor. Then, a random rotation tensor, obtained using the algorithm by Arvo [45], is applied to rotate the diagonal orientation tensor, resulting in the final orientation tensor. For planar reference orientation tensor, where only two out of three diagonal components are non-zero, the process involves sampling only 2 diagonal components associated with the x and y directions. Subsequently, the resulting diagonal matrix is rotated randomly around the z-axis, followed by a 90-degree rotation around randomly selected x, y, or z-axis. For uni-directional reference orientation tensor, it is sampled simply by randomly choosing whether the first, second or third diagonal component equals 1, with all other components being set to 0. More detailed information can be found in [1].

3.2 Random Strain Path Generation

This study utilizes a procedure for generating random reference orientation tensors employed by Friemann et al.[7, 1]. Firstly, the number of drift directions is randomly selected from 1, 2, 5, and 10. Then, a set of strain drift directions is sampled from a normal distribution with a mean of 0 and a standard deviation of 1. 100 time steps are obtained by repeating each strain drift direction multiple times. Additionally, at each time step, a random noise vector, sampled from a normal distribution with a mean of 0 and standard deviation of 1, is added. Subsequently, the noise vector is scaled by a factor γ, ranging from $0 - 1$. Finally, a cumulative sum is calculated for the resulting array and scaled so that the largest component is equal to the prescribed maximum strain ε_{max}, between 0.01 to 0.05. Table 3.2 gives the random strain path generation settings.

Table 3.2: Settings used for random strain path generation

Parameter	**Distribution**	**Range/ Value**
$n - TimeSteps$	-	100
$n - Drift$	uniformly sampled	(1, 2, 5, 10)
γ	uniformly sampled	(0-1)
ε_{max}	uniformly sampled	(0.01-0.05)

The choice of 100 time steps ($n - TimeSteps$) was driven by the computational requirements of RVE simulations. The number of drift directions ($n - Drift$) is randomly selected from a set of options, including 1, 2, 5, or 10. As the number of drift directions increases, the complexity of the strain path also increases. However, it also leads to potential issues for convergence in the simulation. Therefore, a maximum of 10 drift directions are selected to balance the complexity and convergence considerations.

3.3 RVE Size Determination

An RVE should contain enough information about the micro-structure, and it is crucial for it to statistically represent the micro-structure. By including sufficient micro-structure details, the RVE can provide meaningful prediction and accurately capture the material's mechanical response. Therefore, it is essential to conduct a study on the RVE size to ensure about the representativeness of the obtained data.

To determine the RVE size, Mirkhalaf et al. [46] proposed two criteria: (i) the coefficient of variation of the deformation behavior for different RVE realizations with the same size should be less than a desirable value, (ii) the average behavior of all realizations with the same RVE size should fall within a desirable error range compared to the average response of the next RVE size. This approach [46] is followed in this study for RVE size determination. Each tested RVE size has 4 realizations, subjected to monotonic uni-axial stress loading by applying corresponding strain component from 0-0.05. After conducting a statistical analyses, to balance the computational time and maintain the data accuracy, RVE sizes of 4 times the fiber

length in fiber longitudinal direction and 6 times the fiber diameter in fiber transverse direction (4L6D) is selected for uni-directional fiber orientation. For planar fiber orientation, 2 times the fiber length is chosen for the in-plane directions, and 4 times the fiber diameter is selected for the out-of-plane direction (2L4D). With these selections, the coefficient of variation is approximately below 4%, and the difference of the average response compared to the next RVE size is below 6%. For 3D fiber orientation distributions, the maximum RVE size is limited to 1.875 times the fiber length in all directions (1.875L) with the FFT solver. Based on the findings from studies for the uni-directional and planar fiber orientation cases, this size should still provide a reasonable level of representativeness. More detailed information about this statistical analysis is provided in Appendix A.1.
It should be mentioned that both FE and FFT analyses are used in this study. A comparison is performed to ensure about the accuracy of the obtained results. More details about these simulations and comparisons are found in Appendix A.2.

3.4 Post-processing

After finishing each single simulation, the actual orientation tensor, fiber volume fraction, and stress-strain response are extracted. All extracted data, except the stress responses, are placed into an input array, and calculated stresses are inserted into a target output array.

The data were then divided into training, validation, and test data sets, accounting for 80%, 15% and 5% of the data, respectively. Note that the data with the same fiber orientation case and employed solver are grouped together, and the splitting of data is performed within these groups. Afterwards, the split data is combined to form the final training, validation, and test data sets. This procedure ensures that each data set approximately contains the same percentage of data with a certain fiber orientation case and generated by a particular solver. Table 3.3 provides the number of data in the data set for different orientation cases and solvers. Figure

Table 3.3: Number of data samples generated to form the full-field data set.

Fiber orientation case	Solver	Train	Valid	Test	Total
UD	FE	14	2	1	17
	FFT	20	4	1	25
Planar	FE	104	20	6	130
	FFT	100	19	6	125
3D	FE	0	0	0	0
	FFT	200	38	12	250
All		438	83	26	547

3.1 displays a scatter plot representing the principal components of the orientation tensor (a_{11}, a_{22}, and a_{33}) for all generated samples. The color of each point in the scatter plot represents the fiber volume fraction of the sample. It is worth noting that the total amount of data generated is approximately 0.07% of the mean-field data set used to develop the original network. And for the uni-directional fiber orientation

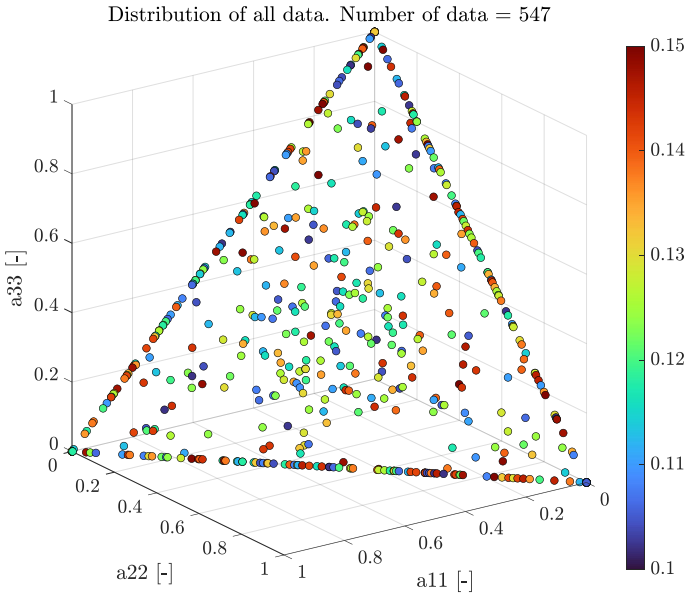


Figure 3.1: Distribution of the generated data in terms of their diagonal orientation tensor components. Different colors represent different fiber volume fractions.

samples, there were 10 samples orientated in the 11 direction, 18 samples in the 22 direction, and 14 samples in the 33 direction. The mean principal components a_{11}, a_{22}, and a_{33} were found to be 0.3366, 0.3262, and 0.3366, respectively. All are close to the expected mean of 1/3, as the property of orientation tensor enforces the sum of the principal components equal to 1 and there is no priority in any direction . Additionally, the mean of the fiber volume fraction is 0.1237, close to the expected mean of 0.125.

4

Network Development

In this chapter, the original network that this study aims to enhance is introduced. Followed by a detailed description of two transfer learning approaches that are experimented with, fine-tuning and correction neural networks. The training parameters and evaluation metrics used to assess neural network performance are also discussed.

4.1 Original Network

Previously, Friemann et al. [7, 1] have developed the original RNN using mean-field simulations. It has 13 inputs including 6 unique orientation tensor components, a fiber volume fraction, and a strain path (a sequence of 6 unique strain tensor components). The inputs were standardized utilizing the Z-score method with the mean and standard deviation calculated from the training data set. The RNN architecture consisted of 3 GRU layers [40], with 500 hidden states in each layer. Following the GRU layers, there is a dropout layer [47] with a 50% dropout rate, and a fully connected layer with 6 neurons. The fully connected layer utilized a linear activation function to output the stress prediction (a sequence of 6 unique stress tensor components). Figure 4.1 illustrated the architecture for the original RNN. The RNN model was developed on a data set consisting of 40,000 samples

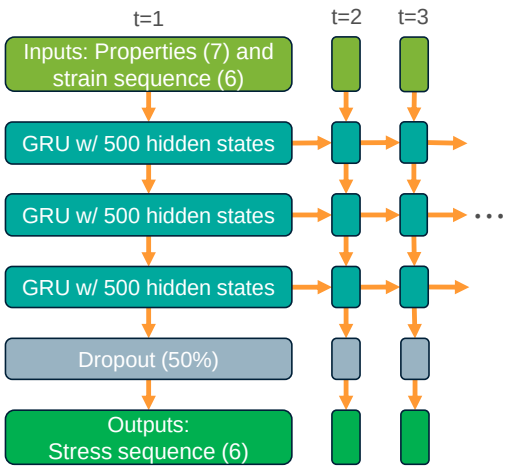


Figure 4.1: Network architecture for the original RNN.

generated using mean-field simulations. A single combination of a matrix and fiber was considered. However, a variety of different fiber orientation tensors and fiber volume fractions were considered. Also, very complex 6D strain paths, with 2,000

time steps, were generated and used. The developed network demonstrates a very good performance, and accurate stress-strain predictions are obtained. However, the obtained accuracy is for low-fidelity mean-field simulations. Throughout the rest of the report, the network trained with mean-field data is referred to as the "original network".

4.2 Transfer Learning Approaches

This section presents the transfer learning approaches investigated to enhance the network performance by leveraging the original network and incorporating a small amount of full-field data.

4.2.1 Fine-tuning

The original network already has the parameters that are trained to recognize the pattern between SFRC properties together with strain path to stress responses in mean-field analysis. Despite the difference in fidelity between mean-field and full-field analysis due to different approaches, both methods are trying to predict the elasto-plastic behavior of SFRC. Therefore, their results should exhibit the same general trends. For the same reason, the original network's architecture is expected to be well-suited for the new but similar task of predicting the full-field analysis stress responses. Additionally, despite the differences in tasks, both networks share the same input and output structure. Therefore, a new network with the same structure can copy all the weights, biases, and input normalization parameters (mean and standard deviation of the input) from the original network. By doing so, the initial position in the optimization landscape would be close to the minimum for the new network. Subsequently, the new network can be trained using high-fidelity full-field data to further approach the minimum. Figure 4.2 shows the network's architecture and how the transfer learning approach is applied.

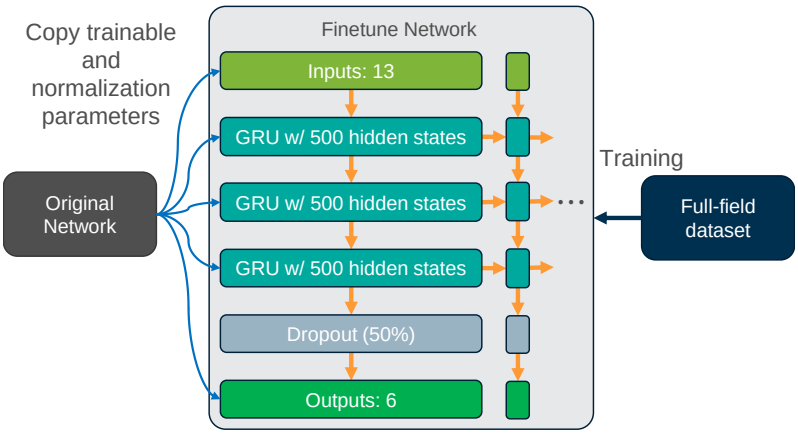


Figure 4.2: Network architecture and the transfer learning approach for fine-tuning network.

4.2.2 Correction Neural Network

Since the difference in mean-field and full-field simulation results is modest, a relatively small correction network and a small full-field data set could be sufficient to find the correlation between mean-field and full-field results. And correcting the outputs of the original network (i.e., predicted mean-field stress result) to a prediction that is close to the full-field stress result. To implement this approach, the original network is duplicated in the new neural network and freezes all trainable parameters by setting the corresponding learning rate factor at 0. Subsequently, a layer normalization [36] is added after the output of the original network to normalize the output before feeding it to the correction network. The correction network is tested using m GRU layers with n hidden states, followed by a dropout layer with a 50% dropout rate to mitigate over-fitting. Finally, a fully connected layer with 6 neurons and linear activation is utilized to output stress prediction. Figure 4.3 shows the network's architecture for the GRU correction network.

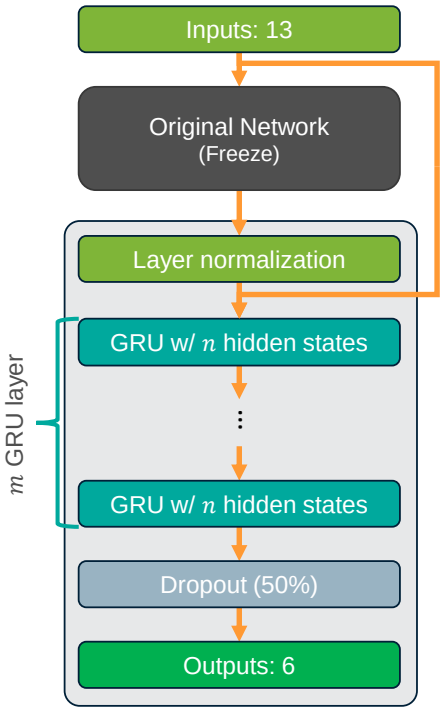


Figure 4.3: Network architecture for the GRU correction network.

4.3 Training Parameters

To train the neural network, a loss function needs to be defined. In this study, the default loss function for time-series regression in Matlab [48] is used:

$$loss = \frac{1}{2S} \sum_{i=1}^{S} \sum_{j=1}^{R} (t_{ij} - O_{ij})^2 \tag{4.1}$$

where S is the sequence length, and R is the number of output, t is the target, and O is the prediction.

ADAM optimizer was employed to minimize the loss function, as it has demonstrated good performance across a wide range of neural networks. The optimizer parameters, such as gradient decay factor β_1, squared gradient decay factor β_2, and denominator offset ε were set to their default values. These default values were chosen as they are generally considered suitable for a wide range of neural networks [43]. Also, L_2-regularization is used to prevent over-fitting [49] and gradient clipping is used to prevent exploding gradient [50].
The hyper-parameters are to be optimized and the range of value is documented in Table 4.1. The majority of training hyper-parameters that will be tuned are related to the learning rate, how it decays during training, and the number of iterations. Although the gradient threshold is also tuned, it is less likely to be a critical parameter for final neural network performance.

Table 4.1: Training hyper-parameters optimize range.

Hyper-parameter	Range
Maximum epochs	[100-800]
Mini batch size	[5-50]
Initial learning rate	[0.0001-0.001]
Learning rate drop period	[10-100]
Learning rate drop factor	[0.9-0.99]
Gradient threshold	[0.9-1.1]

The Bayesian optimization function [51] in Matlab [48] was used to optimize the training parameters. Considering the computational time, the maximum number of trials was set to 65, and the objective is set to minimize the validation loss. Subsequently, the iteration corresponding to the minimum validation loss from the best trial was selected as the final neural network configuration.

4.4 Neural Network Performance Evaluation

This section discusses the evaluation of neural network performance. Firstly, an easily understandable and comparable metric is introduced. Then, the evaluation procedure and testing cases are introduced.

4.4.1 Evaluation Metrics

During the neural network training, the loss function (see Equation (4.1)) is employed to evaluate the network performance. However, the loss function may not be an ideal metric for reviewing the neural network performance, primarily due to its lack of normalization. Therefore, comparing the results between different test samples with varying magnitudes of stress is not justified. To address this issue and enable a proper evaluation of the neural network performance across different test cases, an alternative metric was employed. The equivalent von Mises stress is used to determine the Mean Relative Error (MeRE) and Maximum Relative Error (MaRE). By considering the von Mises stress, which combines multiple components

of the stress tensor into one scalar value with physical meaning, it becomes easier to evaluate the overall performance of the neural network. The von Mises equivalent stress is given by

$$\sigma_V = \sqrt{\frac{1}{2}\left[(\sigma_{11}-\sigma_{22})^2 + (\sigma_{22}-\sigma_{33})^2 + (\sigma_{33}-\sigma_{11})^2\right] + 3\,(\sigma_{12}^2+\sigma_{13}^2+\sigma_{23}^2)}. \quad (4.2)$$

The von Mises stress is calculated at each time step for both the target and network prediction. Subsequently, MeRE and MaRE are calculated by the following equations:

$$\text{MeRE} = \frac{\sqrt{\sum_{t=1}^{T}\left(\sigma_V^t - \hat{\sigma}_V^t\right)^2}}{max(\sigma_V^t)T}, \quad (4.3)$$

and

$$\text{MaRE} = \frac{max(\sigma_V^t - \hat{\sigma}_V^t)}{max(\sigma_V^t)}, \quad (4.4)$$

where σ_V is the target von Mises stress, $\hat{\sigma}_V$ is the predicted one, and the superscript t represents the time step in the sequence.

4.4.2 General Loading Test

To evaluate the performance of the neural network under more standard/typical loading conditions, five loading cases were tested. These included: (i) uni-axial stress in normal (σ_{11}) direction, (ii) uni-axial stress in shear (σ_{12}) direction, (iii) bi-axial stress in two normal ($\sigma_{11}+\sigma_{22}$) directions, (iv) bi-axial stress in normal and shear ($\sigma_{11}+\sigma_{23}$) directions, (v) plain-strain in two normal ($\varepsilon_{11}+\varepsilon_{22}$) directions. Each loading case involves applying the corresponding strain components for a single cyclic loading from 0 to 0.035, then to -0.035, and back to 0. For uni-axial and bi-axial stress loading states, Digimat-FE will calculate the remaining strain components to keep other stress components null. While for plane-strain loading, other strain components are kept at 0. Five different RVEs were generated from distinct random reference orientation tensors. Each RVE was subjected to all five general loading cases. The properties of each RVE are presented in Table 4.2.

Table 4.2: Properties (components of the orientation distribution tensor and fiber volume fraction) of the general test samples.

Sample	a_{11}	a_{22}	a_{33}	a_{12}	a_{13}	a_{23}	v_f
#1	0.0197	0.4926	0.4877	0.0172	-0.0223	0.2963	0.1152
#2	0.5193	0.3062	0.1745	0.0613	-0.1175	-0.1690	0.1320
#3	0.6291	0.2851	0.0858	0.3868	0.0636	0.0443	0.1409
#4	0.7748	0.0946	0.1306	0.0540	0.1060	-0.0576	0.1142
#5	0.4435	0.2642	0.2923	-0.2709	-0.0629	0.0234	0.1314

4.4.3 Multi-cycle Loading Test

To evaluate the network's ability to retain information from long cycles of loading, multi-cycle loading tests were performed. In the multi-cycle tests, six uni-axial stress multi-cycle loading cases were examined. These included: (i) σ_{11}, (ii) σ_{22}, (iii) σ_{33}, (iv) σ_{12}, (v) σ_{23}, and (vi) σ_{13}. Each loading case involved applying their corresponding strain component from 0 to 0.04, then to -0.04, and back to 0, for five cycles. Digimat-FE will calculate the remaining strain components to keep other stress components null. One 3D random RVE was generated, and the properties for this RVE are presented in Table 4.3.

Table 4.3: List of multi-cycle test sample properties.

a_{11}	a_{22}	a_{33}	a_{12}	a_{13}	a_{23}	v_f
0.3224	0.3238	0.3539	-0.006116	-0.009096	-0.008734	0.1200

5
Results

In this chapter, the training process of the neural networks is demonstrated. Then, the fine-tuned network performance in various test cases and a study on the GRU correction network concept is presented.

5.1 Neural Network Training

Initially, a neural network is trained and validated solely utilizing the full-field data set. This is to investigate the necessity and importance of using a transfer learning approach. For simplicity, the architecture of this network is kept the same as the original network, since it is optimized for a similar task. The network, trained from scratch using the full-field data set, is referred to as the "From scratch" network throughout the remainder of this paper. Table 5.1 shows the optimized training hyper-parameters resulted in a validation loss of 99.37 MPa^2 and 18.27 MPa^2 for the "From scratch" and fine-tuned network, respectively.

Table 5.1: Finial hyper-parameters value

Hyper-parameters	From Scratch	Fine-tuned
Maximum epochs	729	343
Mini-batch size	11	40
Initial learning rate	0.001	0.0003
Learning Rate Drop Period	10	23
Learning Rate Drop Factor	0.9772	0.9735
Gradient threshold	1.0906	1.0538

For the network enhancement, an issue is the difference in the number of load increments between mean-field (2,000 steps) and full-field simulations (100 steps). To address this, linear interpolation was used to obtain a full-field data set with 2,000 time steps. Then, this 2,000 time steps full-field data set was used to train another version of both "From scratch" and fine-tuned network. Upon evaluation, a slight improvement in validation loss was observed for the fine-tuned network, while no improvement was observed for the "From scratch" network. Therefore, the final version of the fine-tuned network was trained using the 2,000 time steps full-field data set, whereas the final version of the "From scratch" network was trained using the original 100 time step. Figures 5.1 and 5.2 display the learning curves of these two networks.
Note that in the fine-tuned network training plot, the validation loss appears to be lower than the training loss. This is mainly due to the dropout layer and prior train-

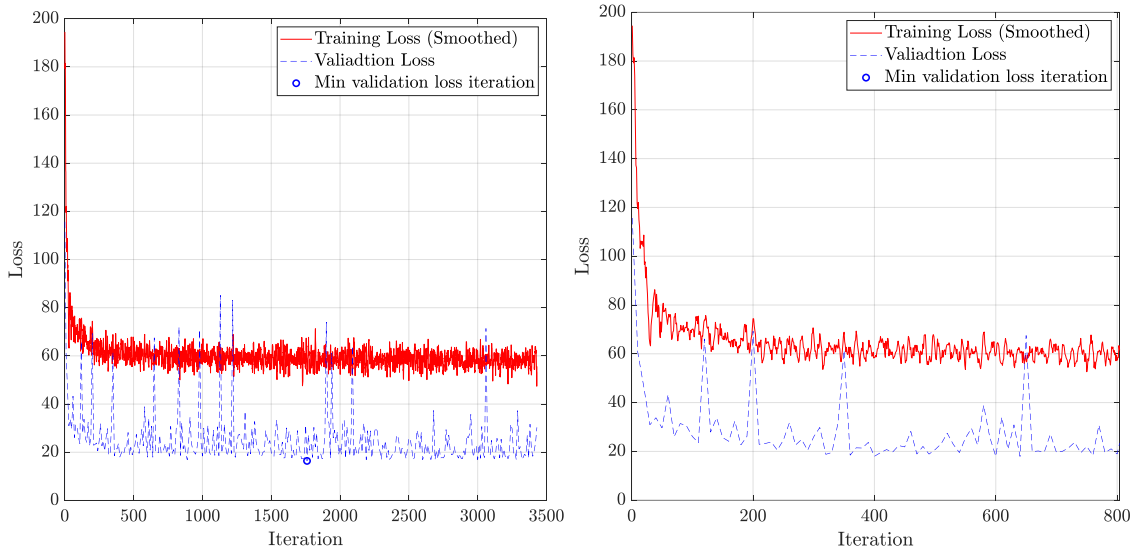


Figure 5.1: The learning curve for the fine-tuned network (left) and zoomed in for the first 800 iterations (right).

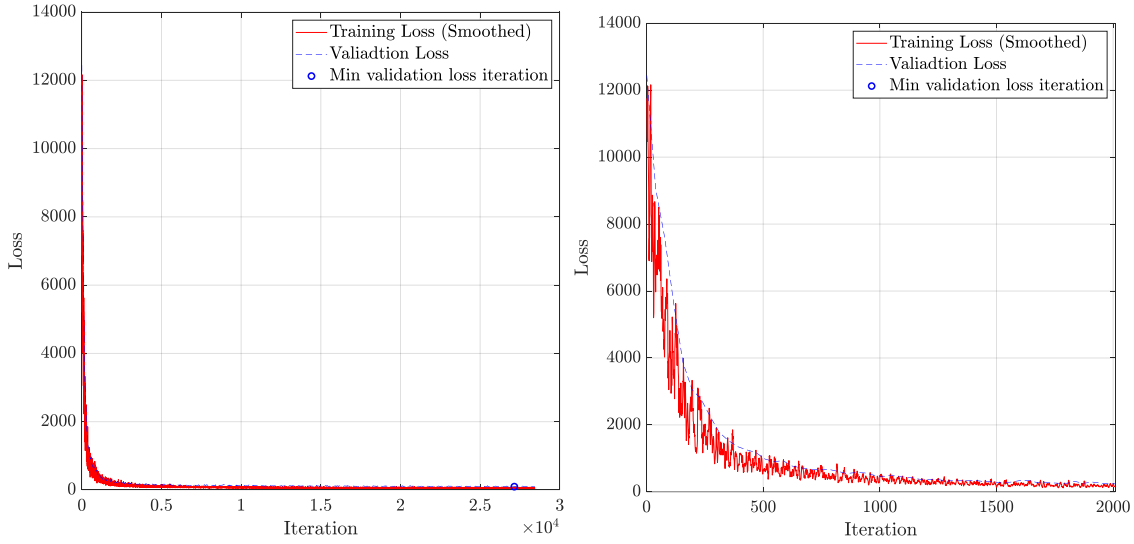


Figure 5.2: The learning curve for the "From scratch" network (left) and zoomed in for the first 2000 iterations (right).

ing with a large mean-field data set. Therefore, the performance can be impacted by the dropout during training, leading to a higher training loss compared to the validation loss. On the other hand, for the "From scratch" network, only a small training data set was used, containing less variety of features. Therefore the dropout effect on training loss is less profound, resulting in a lower training loss compared to validation. Figure 5.3 illustrates the learning curve for the fine-tuned network without the dropout layer, showing a lower training loss than validation loss.

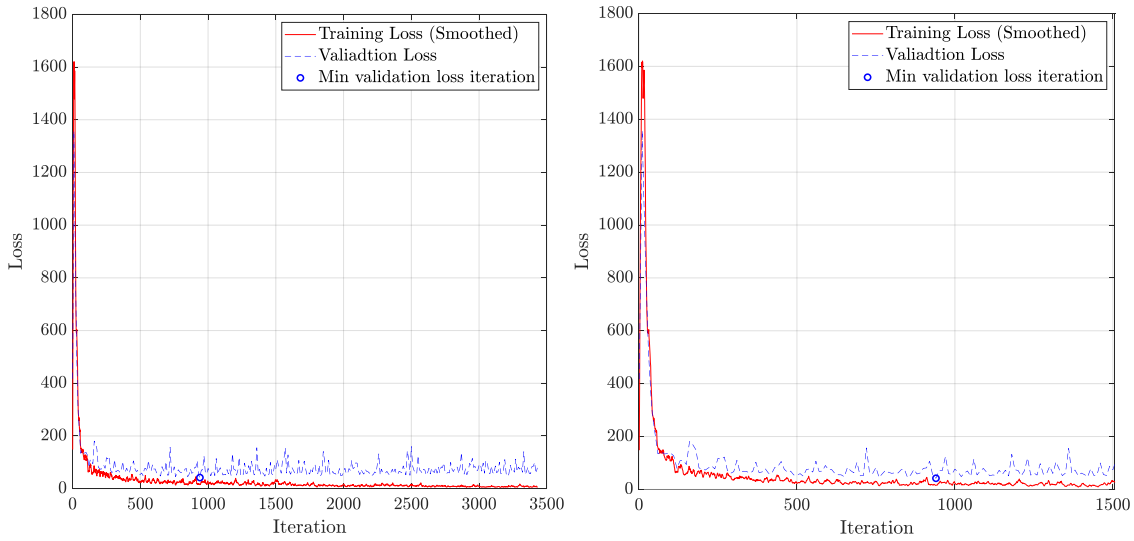


Figure 5.3: The learning curve for the fine-tuned network with dropout layer removed (left) and zoomed in for the first 1500 iterations (right).

5.2 Evaluation procedure

Since different networks are trained utilizing data sets with different time steps, and hence, time steps of the test samples can be a factor when evaluating the networks' performance. To establish a fair performance evaluation, it is reasonable to use the same time steps that the network has trained on for predicting test samples. If the network has trained on a higher number of time steps than the test sample, linear interpolation is applied to the test sample input to allow the network to make predictions with the same time steps used during training. After obtaining the prediction, the intermediate predictions are removed before comparing them to the target. However, if the sample's time steps exceed the time steps that the network has trained on, the test input is fed into the network without any modification. This is the case for testing "From scratch" network on bi-axial stress loading cases, where the samples have 200 time steps, while the network is trained on 100 time steps. Figure 5.4 illustrates the evaluation procedure.

5.3 Test Data Set Results

The average test data set result for the original, "From Scratch" and fine-tuned networks are presented in Table 5.2. In this data set, both MeRE and MaRE values

Table 5.2: MeRE and MaRE results for the test data set.

	Average MeRE	Average MaRE
Original	0.0943	0.2005
From Scratch	0.0659	0.1378
Fine-tuned	0.0312	0.0675

in the original and "From scratch" networks are approximately 3 and 2 times higher

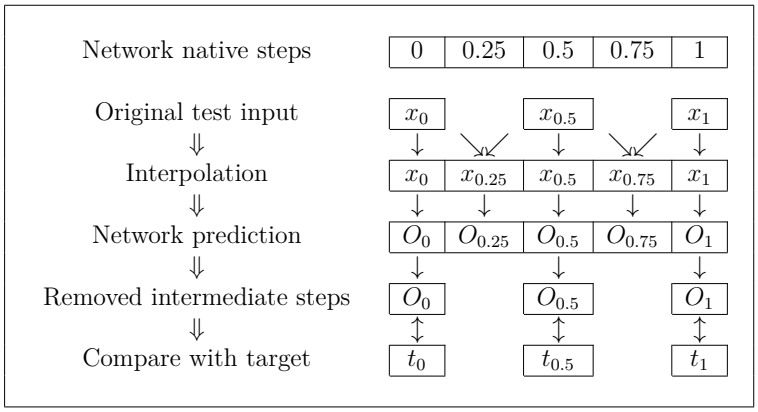


Figure 5.4: Illustration of the testing procedure for a network trained with longer timesteps compared to test data.

than that of the fine-tuned network, respectively. The "From scratch" network also demonstrated better performance compared to the original network in this test. This can be explained by the fact that the test data set consisted of full-field data, whereas the original network was trained on mean-field data. Consequently, there exists a natural divergence between the predictions of the original network and the target values in the test data set.

5.4 General Loading Test Results

The results for the general loading test are presented in Table 5.3. In the general loading test, both average MeRE and MaRE in the original and "From scratch" networks are over 2 and 3.5 times higher, respectively, than that of the fine-tuned network. It is noteworthy that the "From scratch" network performance is worse than the original network, and both MeRE and MaRE are approximately 3 times higher in the general loading test compared to the test data set.

Figure 5.5 shows the stress-strain plot for sample 2 with the uni-axial stress loading. The "From Scratch" network exhibits poor prediction performance, particularly during the unloading phase. On the other hand, the original network generally outperforms the "From Scratch" network. However, inaccuracies often arise during the yielding phase with the non-linear stress-strain relationship. These inaccuracies can be attributed to the original network being trained using mean-field data. Finally, the fine-tuned network demonstrates a significant improvement, not only addressing the weakness of the original network in the yielding phase but also providing more accurate predictions for the unloading phase compared to the "From Scratch" network.

Figure 5.6 shows the result for sample 4 with a plain strain load. Once again, the "From Scratch" network exhibits poor prediction performance, mainly for the shear stress components, σ_{12}, σ_{23} and σ_{13}. Its predictions fail to capture the general trend of the target, indicating the amount of data is insufficient for it to handle different types of loading. On the other hand, the original network is capable of making

Table 5.3: MeRE and MaRE results for the general loading tests.

Sample	Loading Case	Original		From scratch		Fine-tuned	
		MeRE	MaRE	MeRE	MaRE	MeRE	MaRE
#1	Uni-axial (σ_{11})	0.0965	0.2038	0.1919	0.5069	0.0306	0.0760
	Shear (σ_{12})	0.0438	0.0866	0.1967	0.6270	0.0504	0.1270
	Plain strain ($\varepsilon_{11} + \varepsilon_{22}$)	0.1147	0.2474	0.1393	0.3100	0.0950	0.1729
	Bi-axial ($\sigma_{11} + \sigma_{23}$)	0.1105	0.2183	0.2122	0.4184	0.0668	0.1323
	Bi-axial ($\sigma_{11} + \sigma_{22}$)	0.1381	0.2766	0.2353	0.4862	0.0830	0.1681
#2	Uni-axial (σ_{11})	0.1383	0.3075	0.1867	0.4446	0.0409	0.0747
	Shear (σ_{12})	0.1261	0.2545	0.1870	0.3934	0.0377	0.0811
	Plain strain ($\varepsilon_{11} + \varepsilon_{22}$)	0.1498	0.3304	0.1518	0.2743	0.0450	0.0769
	Bi-axial ($\sigma_{11} + \sigma_{23}$)	0.1468	0.2894	0.2312	0.3834	0.0470	0.0994
	Bi-axial ($\sigma_{11} + \sigma_{22}$)	0.1256	0.2669	0.2201	0.4720	0.0439	0.0795
#3	Uni-axial (σ_{11})	0.0946	0.1991	0.2461	0.4894	0.0520	0.0840
	Shear (σ_{12})	0.1566	0.3221	0.1275	0.3122	0.0577	0.1374
	Plain strain ($\varepsilon_{11} + \varepsilon_{22}$)	0.2321	0.3978	0.1797	0.3730	0.1296	0.2695
	Bi-axial ($\sigma_{11} + \sigma_{23}$)	0.0883	0.1850	0.2234	0.4917	0.0442	0.0930
	Bi-axial ($\sigma_{11} + \sigma_{22}$)	0.1043	0.2333	0.2501	0.5427	0.0656	0.1280
#4	Uni-axial (σ_{11})	0.1559	0.3388	0.1676	0.3414	0.0535	0.1005
	Shear (σ_{12})	0.0954	0.1836	0.2077	0.4493	0.0561	0.0957
	Plain strain ($\varepsilon_{11} + \varepsilon_{22}$)	0.1874	0.3670	0.1768	0.3617	0.0641	0.1146
	Bi-axial ($\sigma_{11} + \sigma_{23}$)	0.1493	0.2895	0.2191	0.4088	0.0634	0.1173
	Bi-axial ($\sigma_{11} + \sigma_{22}$)	0.1377	0.2707	0.2038	0.4410	0.0639	0.1109
#5	Uni-axial (σ_{11})	0.0972	0.1827	0.2361	0.5582	0.0618	0.1215
	Shear (σ_{12})	0.1469	0.3134	0.2128	0.4930	0.0305	0.0842
	Plain strain ($\varepsilon_{11} + \varepsilon_{22}$)	0.1610	0.3452	0.2084	0.3827	0.0443	0.0889
	Bi-axial ($\sigma_{11} + \sigma_{23}$)	0.0951	0.1841	0.2151	0.3890	0.0607	0.1483
	Bi-axial ($\sigma_{11} + \sigma_{22}$)	0.1136	0.2620	0.2418	0.5628	0.0546	0.1001
Average		0.1282	0.2622	0.2027	0.4365	0.0577	0.1153

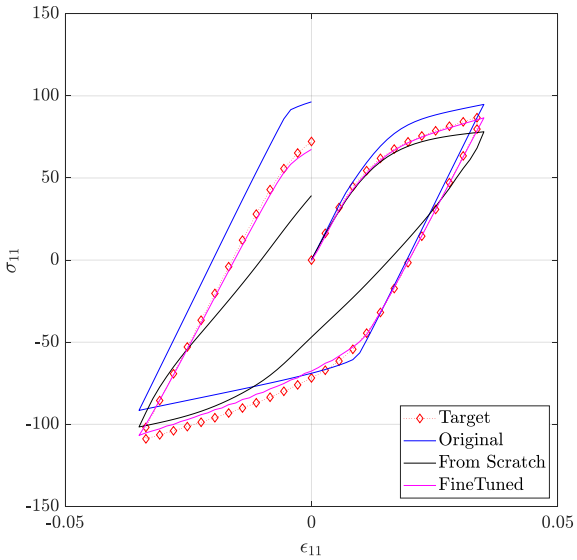


Figure 5.5: Stress-strain plot for sample 2 under uni-axial stress results for the original, "From scratch" and fine-tuned networks.

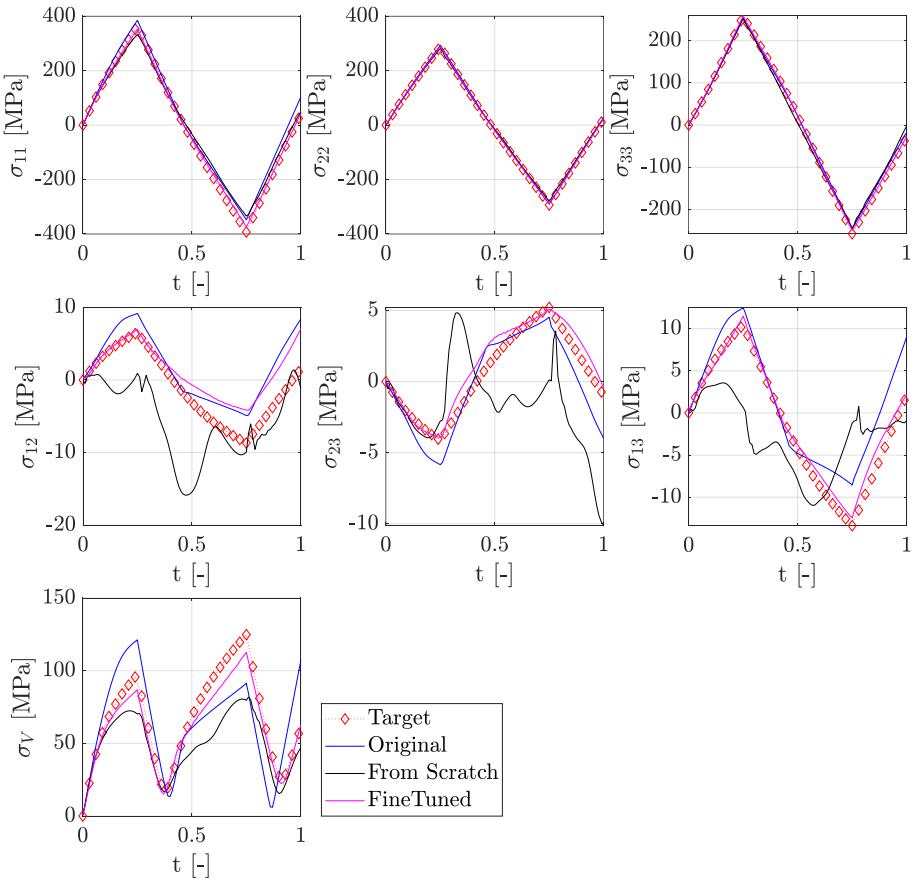


Figure 5.6: Stress results for sample 4 under a plain-strain ($\varepsilon_{11} + \varepsilon_{22}$) loading for the original, "From scratch" and fine-tuned networks.

predictions that follow the trend of the target, although it tends to over or under-estimate the stress values. While the fine-tuned network is able to improve upon the original network and provide the most accurate prediction.

5.5 Multi-cycle Loading Test Results

The MeRE and MaRE results for the multi-cycle loading test are presented in Figure 5.7 and 5.8, respectively. For the original network, a reduction in MaRE and MeRE could be observed from 1 to 3 cycles of loading, and from 3 to 5 cycles, the error increases slightly for each cycle. However, in general, the error does not fluctuate significantly with one to five loading cycles. For the "From Scratch" network, the MeRE and MaRE were the highest in all loading cases and all numbers of cycles. Moreover, a clear increase in error could be observed when the number of cycles increases. For the fine-tuned network, it has the smallest error for one to two cycles. However, the error increases with the number of cycles and has a similar value as the original network for three to five cycles. Nonetheless, it is impressive that the fine-tuned network is also capable of maintaining a relatively low error for a higher number of cycles.

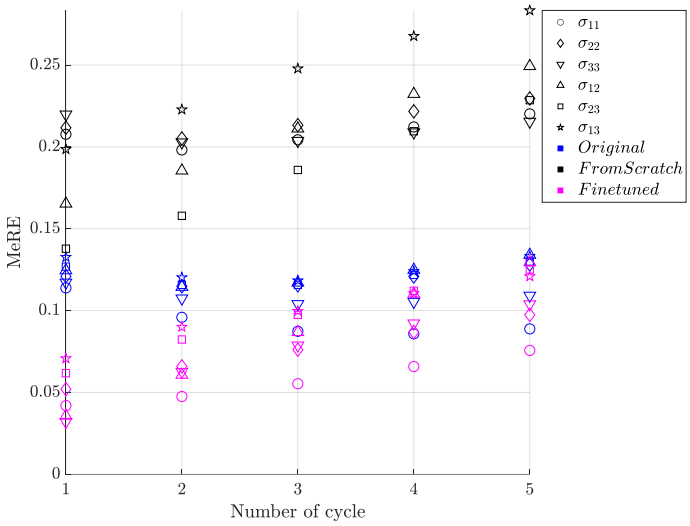


Figure 5.7: MeRE result for cyclic loading test.

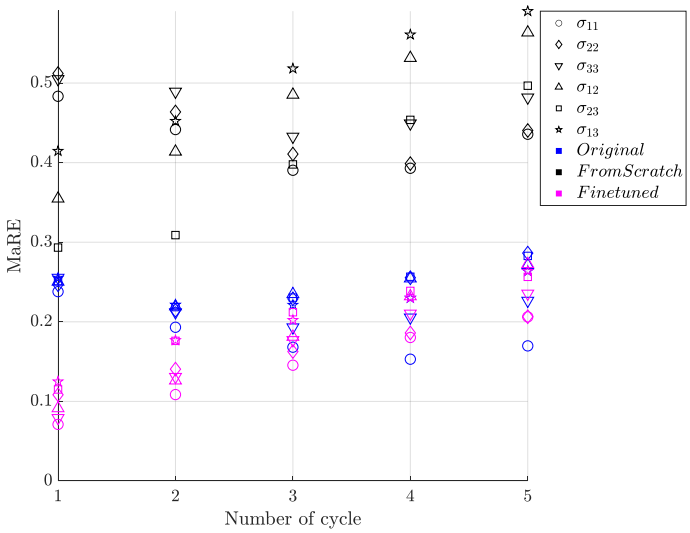


Figure 5.8: MaRE result for cyclic loading test.

5.6 GRU Correction Neural Network Results

Different GRU correction neural network architectures (different numbers of GRU layers and hidden units) have been experimented with and evaluated using a test data set. Figure 5.9 displays the results for the test data set with different setups. Both MeRE and MaRE reduce exponentially by increasing the trainable parameter of the correction network. The minimum MeRE and MaRE attained are 0.0766 and 0.1564, respectively, similar to the "From Scratch" network. It is achieved with 3777006 trainable parameters which correspond to 3 GRU layers with 500 hidden units, the same as the "From Scratch" network's architecture. After this setup, no improvement can be observed by adding more hidden units to the network.

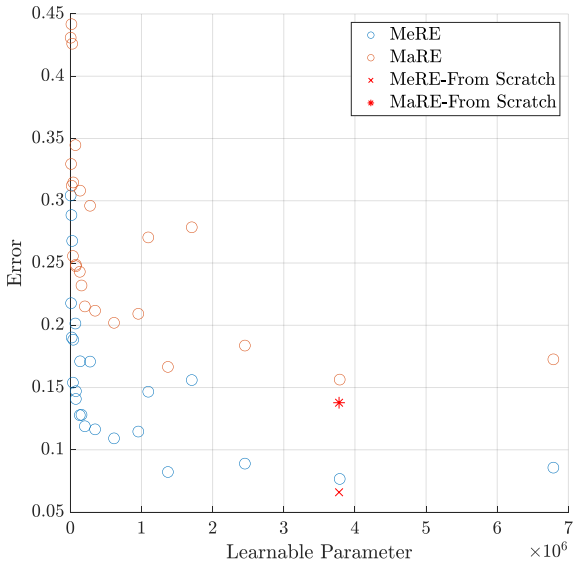


Figure 5.9: Different GRU correction neural networks structure MeRE and MaRE results for the test data set.

6
Discussion and Conclusion

For the fine-tuning approach, both the test data set results and general loading test results have demonstrated a significant performance gain of the fine-tuned network over "From scratch" and original networks. The general loading test further highlighted the necessity and benefits of using the transfer learning approach. When handling loading conditions that are absent from the training data, a large amount of training data is required to provide sufficient information for the network to properly handle these cases. Since the "From scratch" network is trained on a very small full-field data set, its performance is much worse than the fine-tuned network and even the original network. On the other hand, the fine-tuned network was first trained with a significantly larger mean-field data set before fine-tuning with a small full-field data set, enabling it to handle unseen loading conditions with highly accurate predictions. The multi-cycle loading test demonstrated that the "From scratch" that solely relies on less complex full-field data has poor performance for long and complex loading cycles. In contrast, the original network which was developed with highly complex mean-field data shows a steady performance in this test. The fine-tuned network was first trained with the complex mean-field data has helped it to maintain a lower error with a more complex loading cycle.

For the GRU correction network approach, results indicate it fails to effectively leverage the original network to improve accuracy. This can be due to the insufficient amount of data for finding the correlation, the network structure being unsuitable to perform correction, or sub-optimal training hyper-parameters. Nevertheless, the fine-tuning approach is both simpler and has shown a successful transfer of knowledge. Therefore, the correction network approach is not necessary.

Using a neural network for modeling SFRC has a few advantages over a traditional full-field method. First, as the neural network uses the orientation tensor and volume fraction as inputs, SFRC can be modeled with desired properties and a full-field level of accuracy without a trial-and-error process of generating a proper RVE realization. Furthermore, the neural network uses weights and biases to store the learned knowledge, significantly reducing the amount of memory and computational time needed. For example, a full-field simulation requires approximately 13,873 MB of graphics memory and 50 minutes to complete (1.875L 3D RVE FFT analysis running on NVIDIA RTX A4500). In contrast, the size of the neural network only takes up 13.4 MB, and a prediction can be completed within 1 second.

Despite the great performance of the developed network, there are also limitations

to be addressed. Firstly, the current deterministic network is unable to address uncertainties. Potential uncertainties could be attributed to the different responses of different RVE realizations, and existing uncertainties in materials properties (such as properties of natural fibers). There are different possible approaches to capture uncertainties in an RNN, including Bayesian Recurrent Neural Networks (BRNN) (see e.g., [52]) or approximating BRNN using Monte Carlo dropout for RNN [53, 54]. Furthermore, although a variety of orientation tensors and fiber volume fractions are considered, the versatility of the network is limited due to the fixed constitutive properties of matrix and fibers. A more comprehensive model should consider a variety of matrix and fiber properties.

To conclude, this study demonstrates a transfer learning approach to efficiently deploy expensive high-fidelity data to develop a highly accurate recurrent neural network for predicting elasto-plastic responses of SFRCs. The network is developed by leveraging an original network trained with a large amount of low-fidelity mean-field data and fine-tuned with a small amount of high-fidelity full-field data. This approach shows high performance in predicting high-fidelity full-field results, and an extraordinary performance gain when predicting previously unseen loading conditions, compared to the network trained solely on a small full-field data set. This greatly reduces the size of the full-field data set required to develop a network with such performance. Finally, the proposed approach eliminates the trial-and-error process of RVE generation, and predictions can be completed within seconds, thus avoiding the drawbacks of full-field simulation.

Bibliography

[1] J Friemann, B Dashtbozorg, M Fagerström, and SM Mirkhalaf. A micromechanics-based recurrent neural networks model for path-dependent cyclic deformation of short fiber composites. *International Journal for Numerical Methods in Engineering*, 124(10):2292–2314, 2023, doi:10.1002/nme.7211.

[2] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[3] Arthur Robinson, Alan Taub, and Gregory Keoleian. Fuel efficiency drives the auto industry to reduce vehicle weight. *MRS Bulletin*, 44:920–923, 12 2019, doi:10.1557/mrs.2019.298.

[4] Richard Stewart. Automotive composites offer lighter solutions. *Reinforced Plastics*, 54(2):22–28, 2010, doi:https://doi.org/10.1016/S0034-3617(10)70061-8.

[5] Mark Holmes. Global carbon fibre market remains on upward trend. *Reinforced Plastics*, 58(6):38–45, 2014, doi:https://doi.org/10.1016/S0034-3617(14)70251-6.

[6] Mark Holmes. Recycled carbon fiber composites become a reality. *Reinforced Plastics*, 62(3):148–153, 2018, doi:https://doi.org/10.1016/j.repl.2017.11.012.

[7] J. Friemann, B. Dashtbozorg, Martin Fagerström, and Mohsen Mirkhalaf. Predicting the elasto-plastic response of short fiber composites using deep neural networks trained on micro-mechanical simulations. In *VIII Conference on Mechanical Response of Composites*, 01 2021.

[8] P. Kanouté, Daniela Boso, J. Chaboche, and Bernhard Schrefler. Multiscale methods for composites: A review. *Archives of Computational Methods in Engineering*, 16:31–75, 03 2009, doi:10.1007/s11831-008-9028-8.

[9] Kevin Breuer and Markus Stommel. Prediction of short fiber composite properties by an artificial neural network trained on an rve database. *Fibers*, 9:8, 02 2021, doi:10.3390/fib9020008.

[10] F. Ghavamian and A. Simone. Accelerating multiscale finite element simulations of history-dependent materials using a recurrent neural network. *Computer Methods in Applied Mechanics and Engineering*, 357:112594, 2019, doi:https://doi.org/10.1016/j.cma.2019.112594.

[11] Xin Liu, Su Tian, Fei Tao, and Wenbin Yu. A review of artificial neural networks in the constitutive modeling of composite materials. *Composites Part B: Engineering*, 224:109152, 2021, doi:https://doi.org/10.1016/j.compositesb.2021.109152.

[12] Fasikaw Kibrete, Tomasz Trzepieciński, Hailu Shimels Gebremedhen, and Dereje Engida Woldemichael. Artificial intelligence in predicting mechanical properties of composite materials. *Journal of Composites Science*, 7(9), 2023, doi:10.3390/jcs7090364.

[13] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010, doi:10.1109/TKDE.2009.191.

[14] N. Mentges, B. Dashtbozorg, and S.M. Mirkhalaf. A micromechanics-based artificial neural networks model for elastic properties of short fiber composites. *Composites Part B: Engineering*, 213:108736, 2021, doi:https://doi.org/10.1016/j.compositesb.2021.108736.

[15] Ling Wu, Van Dung Nguyen, Nanda Gopala Kilingar, and Ludovic Noels. A recurrent neural network-accelerated multi-scale model for elasto-plastic heterogeneous materials subjected to random cyclic and non-proportional loading paths. *Computer Methods in Applied Mechanics and Engineering*, 369:113234, 2020, doi:https://doi.org/10.1016/j.cma.2020.113234.

[16] M. Mozaffar, R. Bostanabad, W. Chen, K. Ehmann, J. Cao, and M. A. Bessa. Deep learning predicts path-dependent plasticity. *Proceedings of the National Academy of Sciences*, 116(52):26414–26420, December 2019, doi:10.1073/pnas.1911815116.

[17] Jiyoung Jung, Yongtae Kim, Jinkyoo Park, and Seunghwa Ryu. Transfer learning for enhancing the homogenization-theory-based prediction of elasto-plastic response of particle/short fiber-reinforced composites. *Composite Structures*, 285:115210, 2022, doi:https://doi.org/10.1016/j.compstruct.2022.115210.

[18] Digimat 2021.3 - MF user's guide (PDF).

[19] Suresh G. Advani and III Tucker, Charles L. The Use of Tensors to Describe and Predict Fiber Orientation in Short Fiber Composites. *Journal of Rheology*, 31(8):751–784, 11 1987, doi:10.1122/1.549945.

[20] L. Wu, L. Adam, and L. Noels. A micromechanics-based inverse study for stochastic order reduction of elastic ud fiber reinforced composites analyses. *International Journal for Numerical Methods in Engineering*, 115(12):1430–1456, 2018, doi:https://doi.org/10.1002/nme.5903.

[21] T Mori and K Tanaka. Average stress in matrix and average elastic energy of materials with misfitting inclusions. *Acta Metallurgica*, 21(5):571–574, 1973, doi:https://doi.org/10.1016/0001-6160(73)90064-3.

[22] John Douglas Eshelby. The determination of the elastic field of an ellipsoidal inclusion, and related problems. *Proceedings of the royal society of London. Series A. Mathematical and physical sciences*, 241(1226):376–396, 1957.

[23] A.R. Melro, P.P. Camanho, and S.T. Pinho. Generation of random distribution of fibres in long-fibre reinforced composites. *Composites Science and Technology*, 68(9):2092–2102, 2008, doi:https://doi.org/10.1016/j.compscitech.2008.03.013.

[24] L. Bouaoune, Y. Brunet, A. El Moumen, T. Kanit, and H. Mazouz. Random versus periodic microstructures for elasticity of fibers reinforced composites. *Composites Part B: Engineering*, 103:68–73, 2016, doi:https://doi.org/10.1016/j.compositesb.2016.08.026.

[25] Klaus-Jürgen Bathe. *Finite element procedures*. Klaus-Jurgen Bathe, 2006.

[26] Wenlong Tian, Lehua Qi, Xujiang Chao, Junhao Liang, and Mingwang Fu. Periodic boundary condition and its numerical implementation algorithm for the evaluation of effective mechanical properties of the composites with complicated micro-structures. *Composites Part B: Engineering*, 162:1–10, 2019, doi:https://doi.org/10.1016/j.compositesb.2018.10.053.

[27] H. Moulinec and P. Suquet. A numerical method for computing the overall response of nonlinear composites with complex microstructure. *Computer Methods in Applied Mechanics and Engineering*, 157(1):69–94, 1998, doi:https://doi.org/10.1016/S0045-7825(97)00218-1.

[28] J. Zeman, T. W. J. de Geus, J. Vondřejc, R. H. J. Peerlings, and M. G. D. Geers. A finite element perspective on nonlinear fft-based micromechanical simulations. *International Journal for Numerical Methods in Engineering*, 111(10):903–926, 2017, doi:https://doi.org/10.1002/nme.5481.

[29] Sergio Lucarini, Manas Upadhyay, and Javier Segurado. Fft based approaches in micromechanics: Fundamentals, methods and applications. *Modelling and Simulation in Materials Science and Engineering*, 30, 10 2021, doi:10.1088/1361-651X/ac34e1.

[30] J. Yvonnet. *Computational Homogenization of Heterogeneous Materials with Finite Elements*. Solid Mechanics and Its Applications. Springer International Publishing, 2019.

[31] Bernhard Mehlig. *Machine Learning with Neural Networks: An Introduction for Scientists and Engineers*. Cambridge University Press, 2021.

[32] Yuanyuan Tian, Mi Shu, and Qingren Jia. *Artificial Neural Network*, pages 1–4. Springer International Publishing, Cham, 2020.

[33] Johannes Lederer. Activation functions in artificial neural networks: A systematic overview. 2021, doi:https://doi.org/10.48550/arXiv.2101.09957.

[34] Miroslav Kubat. Neural networks: a comprehensive foundation by simon haykin, macmillan, 1994, isbn 0-02-352781-7. *The Knowledge Engineering Review*, 13(4):409–412, 1999, doi:10.1017/S0269888998214044.

[35] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[36] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. 2016, doi:https://doi.org/10.48550/arXiv.1607.06450.

[37] R. J. Frank, N. Davey, and S. P. Hunt. Time series prediction and neural networks. *Journal of Intelligent and Robotic Systems*, 31(1):91–103, May 2001, doi:10.1023/A:1012074215150.

[38] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. *Dive into Deep Learning*. 2023.

[39] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997, doi:10.1162/neco.1997.9.8.1735.

[40] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase repre-

sentations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[41] Ruo-Yu Sun. Optimization for deep learning: An overview. *Journal of the Operations Research Society of China*, 8(2):249–294, Jun 2020, doi:10.1007/s40305-020-00309-6.

[42] Sebastian Ruder. An overview of gradient descent optimization algorithms. 2017, doi:https://doi.org/10.48550/arXiv.1609.04747.

[43] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[44] S. Kammoun, I. Doghri, L. Adam, G. Robert, and L. Delannay. First pseudo-grain failure model for inelastic composites with misaligned short fibers. *Composites Part A: Applied Science and Manufacturing*, 42(12):1892–1902, 2011, doi:https://doi.org/10.1016/j.compositesa.2011.08.013.

[45] James Arvo. Iii.4 - fast random rotation matrices. In DAVID KIRK, editor, *Graphics Gems III (IBM Version)*, pages 117–120. Morgan Kaufmann, San Francisco, 1992.

[46] S.M. Mirkhalaf, F.M. Andrade Pires, and Ricardo Simoes. Determination of the size of the representative volume element (rve) for the simulation of heterogeneous polymers at finite strains. *Finite Elements in Analysis and Design*, 119:30–44, 2016, doi:https://doi.org/10.1016/j.finel.2016.05.004.

[47] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, jan 2014, doi:https://dl.acm.org/doi/10.5555/2627435.2670313.

[48] The MathWorks Inc. Matlab version: 9.13.0 (r2022b), 2022.

[49] Xue Ying. An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, 1168(2):022022, feb 2019, doi:10.1088/1742-6596/1168/2/022022.

[50] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, page III–1310–III–1318, 2013.

[51] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, 2012.

[52] Meire Fortunato, Charles Blundell, and Oriol Vinyals. Bayesian recurrent neural networks. 2019, doi:https://doi.org/10.48550/arXiv.1704.02798.

[53] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Confer-*

ence on Machine Learning, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR.

[54] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. 2016, doi:https://doi.org/10.48550/arXiv.1512.05287.

A

Appendix 1

A.1 Representative Volume Element Size Determination

Figure A.1 displays examples of Uni-Directional (UD), planar, and 3D fiber orientation RVEs and respective sizes in different directions. The RVE's dimensions are

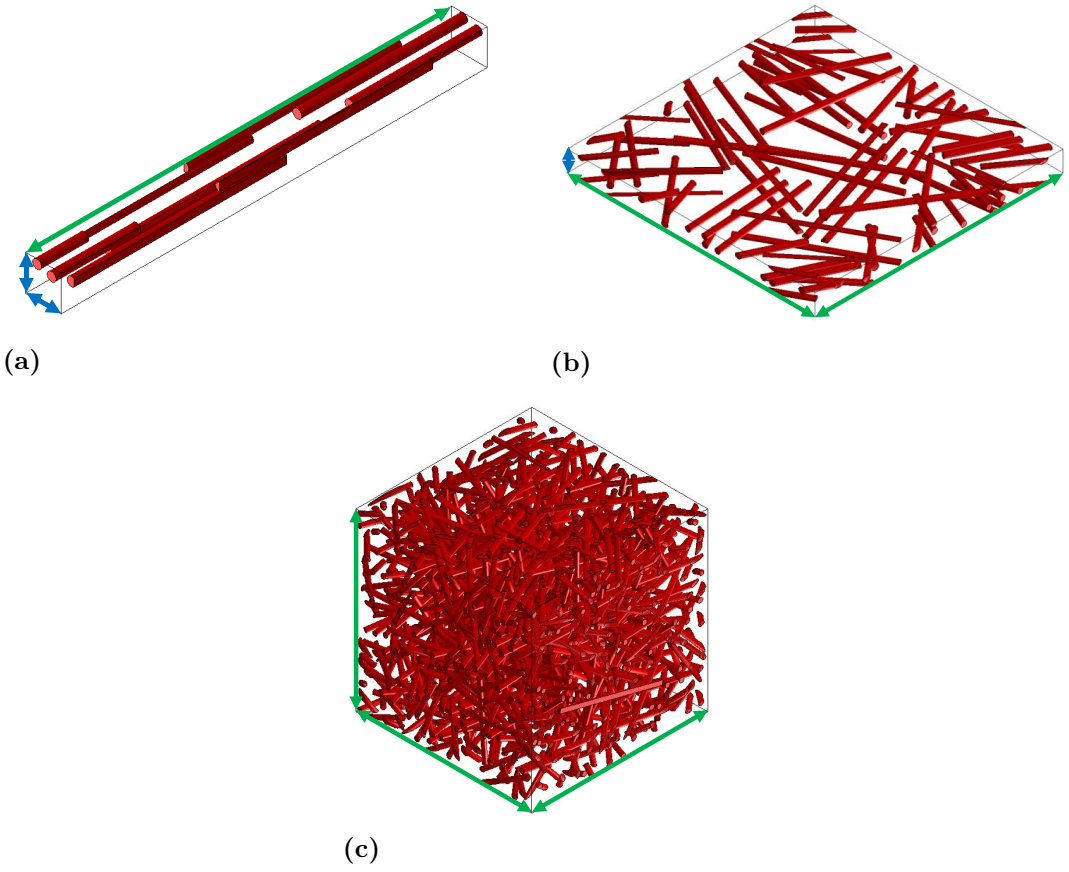


Figure A.1: Represenation of (a) uni-directional, (b) planar, and (c) 3D fiber orientation RVEs.

chosen based on fiber length and diameter, and are indicated with different arrows in different directions.

A.1.1 Uni-directional Fiber Orientation Case

For the uni-directional fiber orientation case, different sizes of micro-structural samples were examined. These sizes were chosen with respect to the fiber dimensions. Specifically, the sizes were chosen as 2, 3, and 4 times the fiber length in the fiber longitudinal direction, and 4, 5, and 6 times the fiber diameter in the transverse direction. The mentioned sizes are referred to as 2L4D, 3L5D, and 4L6D, respectively. Four realizations were generated for each size and the properties for each realization are listed in Table A.1. For each realization, 4 monotonic uni-axial stress loading

Table A.1: List of UD fiber orientation sample properties.

Size	Sample	a_{11}	a_{22}	a_{33}	a_{12}	a_{13}	a_{23}	v_f
2L4D	#1, 2, 3, 4	1	0	0	0	0	0	0.1227
3L5D	#1, 2, 3, 4	1	0	0	0	0	0	0.1361
4L6D	#1, 2, 3, 4	1	0	0	0	0	0	0.1309

cases (with strain range of 0 to 0.05) were tested, namely σ_{11}, σ_{22}, σ_{12} and σ_{23}. The normal stress components demonstrated a higher discrepancy between different realizations, and therefore, the following part focuses on the normal stress results. Figure A.2 displays the average response with bounds of standard deviation, and the coefficient of variation results for uni-axial stress loading in σ_{11} and σ_{22}. Since all

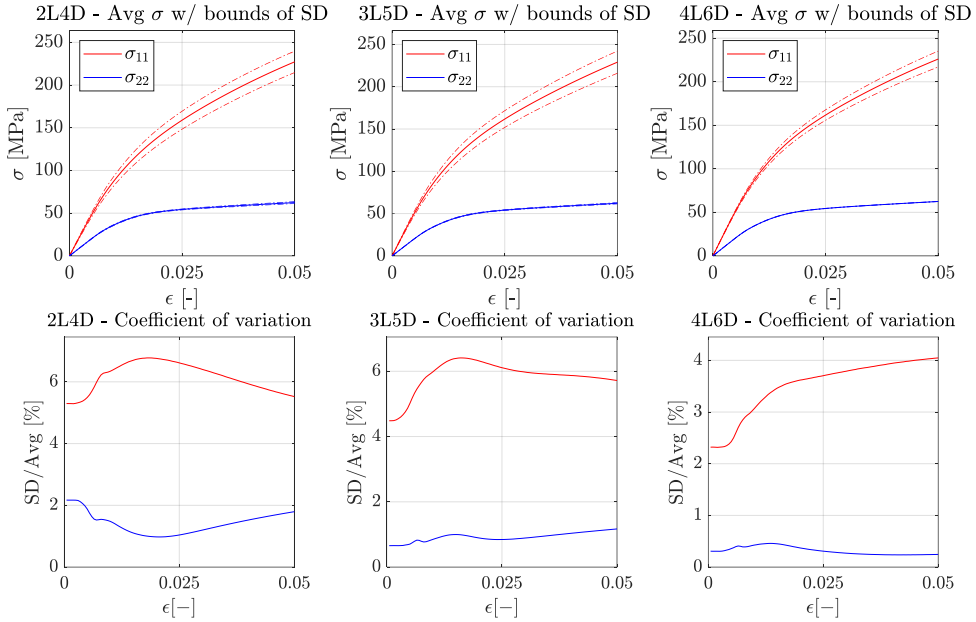


Figure A.2: The average σ_{11}, σ_{22} (top) and coefficient of variation (bottom) results for uni-directional fiber orientation with different RVE sizes. The first, second, and third column correspond to 2L4D, 3L5D, and 4L6D, respectively.

fibers are orientated towards and reinforcing the x-direction, the stress component σ_{11} is considerably higher than σ_{22}. Consequently, the variation in σ_{11} is also higher compared to σ_{22}. However, the σ_{11} variation for all three sizes of micro-structural samples remains below 7%, and below approximately 4% for 4L6D. And Figure

A.3 illustrates the absolute percentage change in average stress when comparing a smaller RVE size to a larger RVE size. The highest change is observed for σ_{11} when comparing 2L4D and 3L5D RVE sizes. However, the change is lower than 5%. And note that the change could also be influenced by the slight variations in fiber volume fractions between samples.

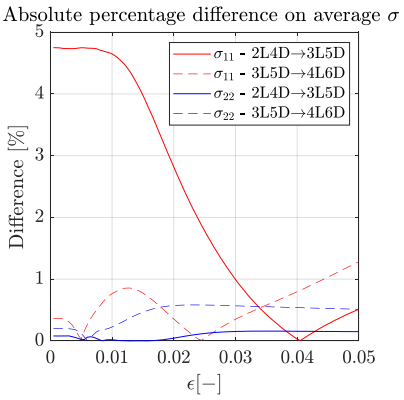


Figure A.3: Absolute percentage different for uni-directional fiber orientation with different RVE sizes.

Based on the analysis, all three RVE sizes appear to be suitable for a uni-directional fiber orientation case. Considering the computational cost associated with these sizes are still manageable, RVE size of 4L6D is selected for the uni-directional fiber orientation sample to achieve maximum accuracy.

A.1.2 Planar Fiber Orientation Case

For the planar fiber orientation case, two sizes for the micro-structural samples were examined, namely 2L4D and 3L3D. Similarly to the UD RVEs, 4 realizations were generated for each size. An attempt is made to generate planar random fiber orientations. Components of the fiber orientation tensor and fiber voluem fractions for different sizes and different relaizations are found in Table A.2. Uni-axial stress

Table A.2: List of planar fiber orientation samples properties.

Size	Sample	a_{11}	a_{22}	a_{33}	a_{12}	a_{13}	a_{23}	v_f
2L4D	#1	0.5141	0.4859	0	0.03666	0	0	0.08001
	#2	0.4863	0.5137	0	0.03701	0	0	0.07999
	#3	0.4963	0.5037	0	-0.04389	0	0	0.08000
	#4	0.507	0.493	0	-0.01167	0	0	0.08000
3L3D	#1	0.5003	0.4997	0	-0.04302	0	0	0.08001
	#2	0.5188	0.4812	0	0.09735	0	0	0.08000
	#3	0.5174	0.4826	0	0.05389	0	0	0.08000
	#4	0.516	0.484	0	-0.09783	0	0	0.08000

loading cases, including σ_{11}, σ_{33}, σ_{12} and σ_{13} were tested, using the same loading method as the uni-directional fiber orientation study. Figure A.4 shows the average response with bounds of standard deviation and the coefficient of variation results for

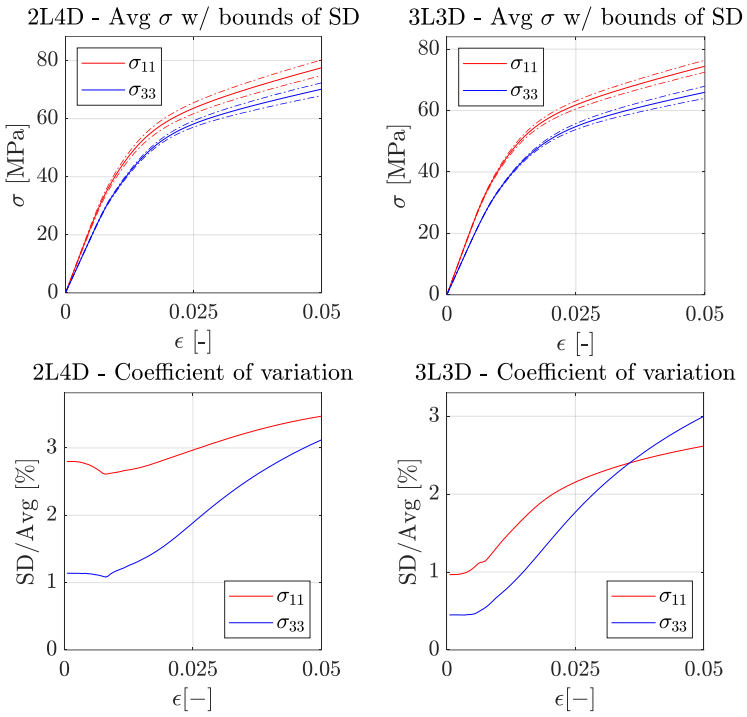


Figure A.4: The average σ_{11}, σ_{33} (top) and coefficient of variation (bottom) results for planar random fiber orientation with different RVE sizes. The first, and second column correspond to 2L4D, and 3L4D, respectively.

uni-axial stress loading in σ_{11} and σ_{33}. Due to the planar nature of fiber distributions, σ_{11} and σ_{33} components are rather close to each other, as it can be clearly seen in Figure A.4. Additionally, it is observed that the variations for the two sizes are lower than 3.5%. Figure A.5 illustrates the absolute percentage of difference in average stress for different sizes of the micro-structural samples. The stress component with the highest change is σ_{33}, with a maximum of just below 6%.

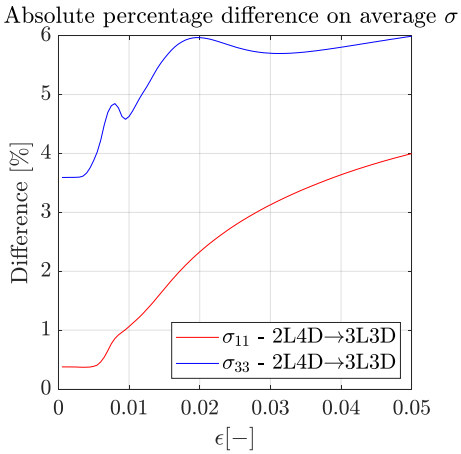


Figure A.5: Absolute percentage different for planar random fiber orientation with different RVE sizes.

Considering the low variation observed with the 2L4D size and relatively low changes

in average behaviour compared to the 3L3D size, using the 2L4D size should be able to provide representative results while still allowing for a manageable computational time. Therefore, an RVE size of 2L4D is selected for the planar fiber orientation sample.

A.2 Finite Element and Fast Fourier transfer Simulations

FE and FFT solvers within Digimat-FE were employed to conduct the simulations. The FE solver is used with a conforming mesh, and the FFT solver utilizes a grid. Figure A.6 illustrates the different discretization methods. The conforming mesh

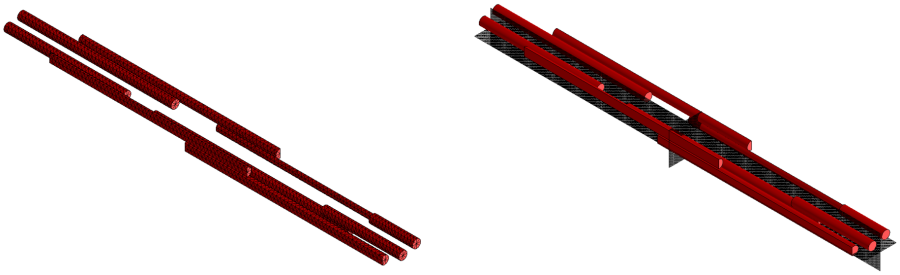


Figure A.6: Fibers spatially discretized using a conforming mesh for the FE solver (left) and the grid used by the FFT solver (right).

preserves a higher accuracy of geometrical characteristic of the original RVE realization. This results in a more accurate prediction. On the other hand, the FFT solver discretizes the domain using a grid, which may potentially influence the results. Therefore, a comparison between the two solvers is conducted using a 3D RVE subjected to a random strain path. The RVE properties are given in Table A.3. Figure A.7 shows a comparison of the obtained results using the two methods.

Table A.3: Properties of a 3D RVE sample used for solver comparison.

Size	a_{11}	a_{22}	$a33$	a_{12}	a_{13}	a_{23}	v_f
1.67L	0.2943	0.3842	0.3215	-0.007636	0.01868	0.03679	0.1111

Note that the FE analysis was aborted at increment Number 57, while the FFT analysis was able to complete the simulation.

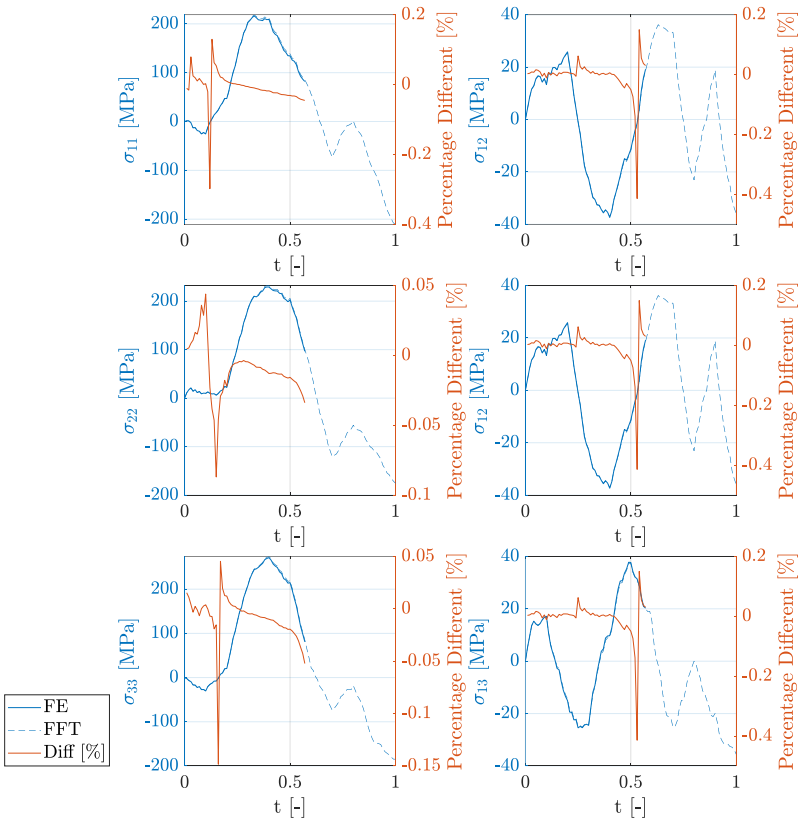


Figure A.7: A comparison of results obtained using FE and FFT methods for a 3D RVE subjected to a random loading path.

It is observed that the stress responses calculated from FE and FFT analyses are quite similar. The maximum percentage of the absolute difference between the two approaches is less than 0.5%. Therefore, for the purpose of modeling elasto-plastic response of SFRCs, it is possible to use the FFT approach for accurate high-fidelity full-field simulations.

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY