國立臺灣大學工學院土木工程學系

碩士論文

Department of Civil Engineering

College of Engineering

National Taiwan University

Master Thesis

# 圖增強的深層材料網絡：泛用微結構的多尺度材料模擬

# Graph-enhanced Deep Material Network: Multiscale Materials Modeling with Microstructural Generalization

蔣斯柏

Jimmy Gaspard Jean

指導教授: 陳俊杉 博士

Advisor: Chuin-Shan Chen, Ph.D.

中華民國 112 年 6 月

June, 2023

doi:10.6342/NTU202301463

# Acknowledgements

I would like to express my sincere gratitude to my esteemed advisor Prof. Chuin-Shan (David) Chen for having granted me the opportunity to join the Computational Mechanics and Materials AI (CMMAi) group as an undergraduate intern in the summer of 2020, and later for my Master's degree in 2021. His unwavering guidance and support, expertise, and insightful feedbacks during the past three (3) years have been essential in shaping the direction and quality of this research.

I am also indebted to Dr. Tung-Huan Su (Michael) for mentoring me from when I started as an intern in the CMMAi group until now, as I am completing my Master's degree. His continued support and encouragement have propelled me to perform well. Our usual pre-meeting and post-meeting discussions have been instrumental in facilitating the progress of this research.

I would also like to thank Szu-Jui Huang who assisted me in debugging codes, and in the derivation of formulas related to the deep material network. He is also the author of the codes used for generating the microstructures presented in this work. I also want to thank Berry Wei who joined our research on the deep material network and helped me check the Mandarin version of this work's abstract. I would also like to extend my thanks to Dr. C. T. Wu and Prof. Shu-Wei Chang for serving on my thesis committee. Their



doi:10.6342/NTU202301463

insights on this work and suggestions for future directions have truly been valuable.

To my dear mother (Marie Justine Chéry) and father (Falieur Jean), thank you for instilling in me the values of perseverance, discipline, and a thirst for knowledge. Your sacrifices, both big and small, have allowed me to pursue my dreams and achieve this milestone. I dedicate this achievement to both of you, as a testament to your hard work, without which this accomplishment would not have been possible. To my younger brother (Kelly R. Jean) for his support and encouragement. I would also like to thank friends from the International Students group at Grace Baptist Church with whom I have built a great relationship and precious memories during the course of my Master's studies. The list includes Cesar Gonzalez, Regina Pai, Cris Yu, Helina Lao, Praise Tan, Maria José.

Above all, I thank God for intellectual abilities and guiding me as I walk through this earthly journey. I look forward to what lays ahead.



doi:10.6342/NTU202301463

# 摘要

近年來，工業應用逐漸從鋼、鋁等傳統材料轉向纖維增強聚合物等複合材料。人們對複合材料日益增長的興趣源自於其優於傳統材料的性能。為了模擬這些複雜材料的行為並優化其設計，基於有限元方法 (FEM) 以及基於快速傅立葉變換 (FFT) 的多尺度模擬長期以來一直被視為重要工具。然而成本昂貴這一缺點促使人們使用基於機器學習 (ML) 的代理模型來加速材料的多尺度建模。其中深層材料網路（DMN）表現較為突出，因為它一方面能夠將微觀結構的表示 (representation) 壓縮到幾個自由度。另一方面，它只需要在線彈性數據上進行訓練，即可用於預測複雜的非線性材料行為。然而，DMN 受到一次只能對單個代表性體積元素 (RVE) 進行建模的限制。一些過去的研究考慮了通過利用 RVE 的微觀描述符來減輕這種限制的各種方法。但是，他們只解決了部分問題。在本研究中，我們希望從根本上解決這個問題。我們通過利用 RVE 中的全部微觀結構細節來做到這一點。我們提出了一種混合圖神經網絡（GNN）-DMN 模型，它可以單獨處理多個微觀結構並生成它們的小規模 DMN 表示 (representation)。我們通過考慮顆粒增強複合材料的兩個 RVE 系列來證明我們的概念：circular inclusions 與 ellipse-shaped inclusions 。研究結果表明不同微觀結構的建模皆可以在本研究提出的模型得到良好的表現。本研究為進一步研究將具有複雜微觀結構信息泛化 (generalization) 的大型多尺度模型打開了大門，類似於自然語言處理領域的大型語言模型（LLM）。



doi:10.6342/NTU202301463

**關鍵字**：計算力學、多尺度建模、複合材料、圖神經網絡、深層材料網絡、基於圖的力學深度學習



doi:10.6342/NTU202301463

# Abstract

In recent years, there has been a progressive shift in various industries from traditional materials such as steel, aluminum to composite materials like fiber-reinforced polymers. These industries range from the automotive to the construction industry. Such a growing interest in composites has stemmed from their superior performance over traditional materials. To simulate the behavior of these complex materials and optimize their design, finite-element-method (FEM)-based as well as fast Fourier transform (FFT)-based multiscale modeling have long been used as essential tools. However, they suffer the drawback of being computationally expensive. This limitation has given rise to the use of machine learning (ML)-based surrogate models to accelerate the multiscale modeling of materials. One such model that stands out is the deep material network (DMN) due to its ability, on one hand, to compress into a few degrees of freedom the representation of microstructures. On another hand, it only needs to be trained on linear elastic data and yet can be used to predict complex nonlinear material behaviors. Nevertheless, DMN suffers from



doi:10.6342/NTU202301463

the constraint of being able to model a single representative volume element (RVE) at a time. Some works have considered various approaches to alleviate this limitation by leveraging the microscopic descriptors of RVEs. But, they have addressed the issue only in part. In this work, we tackle the issue at its core. We do so by exploiting the entirety of microstructural details in RVEs. We propose a mixed graph neural network (GNN)-DMN model that can single-handedly treat multiple microstructures and generate their small-scale DMN representations. We prove our concept by considering two families of RVEs of particle-reinforced composites: those with circular inclusions and those with ellipse-shaped inclusions. The results show the possibility to unify under one network the modeling of dissimilar microstructures. Our work opens the door for further research towards the development of large multiscale models with generalization capability of microstructural information for heterogeneous material systems, akin to large language models (LLM) in the field of natural language processing.

**Keywords:** Computational mechanics, Multiscale modeling, Composite materials, Graph neural network, Deep material network, Graph-based mechanistic deep learning



doi:10.6342/NTU202301463

# Contents





doi:10.6342/NTU202301463

doi:10.6342/NTU202301463

doi:10.6342/NTU202301463

doi:10.6342/NTU202301463

# List of Figures





doi:10.6342/NTU202301463

doi:10.6342/NTU202301463

doi:10.6342/NTU202301463



doi:10.6342/NTU202301463

# List of Tables





doi:10.6342/NTU202301463



doi:10.6342/NTU202301463

# Chapter 1 Introduction

## 1.1 Background and Motivation

Matter is central to the human experience. The flesh and bones in our body are matter. The homes we inhabit, the cars we drive, the planes we fly, the computers or papers we use for reading are nothing but different end products of materials that have been carefully combined and processed. The history of human civilization and the technological advancements that have accompanied it can, in fact, be seen as the story of mankind's mastery of materials. From the Stone Age, through the Industrial Age, until the current Information Age, our understanding of materials and of their mechanics has progressively increased, thus raising the complexity level of tools we are able to create.

Prominent works in early studies of mechanics include Galileo's *Two New Sciences* [12] published in 1638, and Newton's *Philosophiæ Naturalis Principia Mathematica* [36] published in 1687. The latter provided the basis for the commonly termed "Newtonian mechanics". More recently, the advent of modern computers during the 20th century made possible the birth of computational mechanics, whose origins can be traced back to the 1970s [7]. Since then, various numerical solvers based on methods such as the finite element method (FEM) [15], the Fast Fourier Transform (FFT), have been developed to simulate the behavior of materials. With the increase of computer power and an inter-


doi:10.6342/NTU202301463

est in the scientific community to develop high-performance and specialized materials, multiscale materials modeling has gained in popularity during the last few decades.

Multiscale materials modeling seeks to simulate and link material behavior across a range of scales (e.g., nanoscale, microscale, macroscale). Properties computed at smaller scales are averaged and then integrated into larger scales. One advantage of this approach is that there is no need to describe all the heterogeneities characteristic of a given material at the macroscopic level [53]. In multiscale modeling, the averaging and transfer of properties between scales is usually carried out through computational homogenization. The first works on computational homogenization include the publications of Suquet and others [33, 41], in which the methodology of homogenization has been applied using both FEM and FFT. Nevertheless, its computational burden is not negligeable, even for nowadays most advanced supercomputers. Since then, significant advances in machine learning (ML) have enabled the use of surrogate models to replace FEM/FFT-based methods in computational homogenization. The main advantage of such ML-based surrogate models is their rapidity, hence allowing for faster iterations during material simulation and design [2]. One of the latest of these surrogate models is the deep material network (DMN).

The main feature of DMN is that it only needs to be trained on a small amount of linear elastic data. Yet, it can be used to predict nonlinear material behaviors. As a result, it shows great promise in the future for enabling our understanding and mastery of complex materials. But, the generalization of DMN to multiple microstructures is currently the missing cornerstone for its widespread use in an era of ML-enabled multiscale material modeling and design. This emerging research area is still in its infancy [11, 21, 30]. The current state-of-the-art thus compels us to propose a graph-based framework that can arguably settle, in all aspects, the issue of the generalization of DMN.


doi:10.6342/NTU202301463

## 1.2 Research Objectives

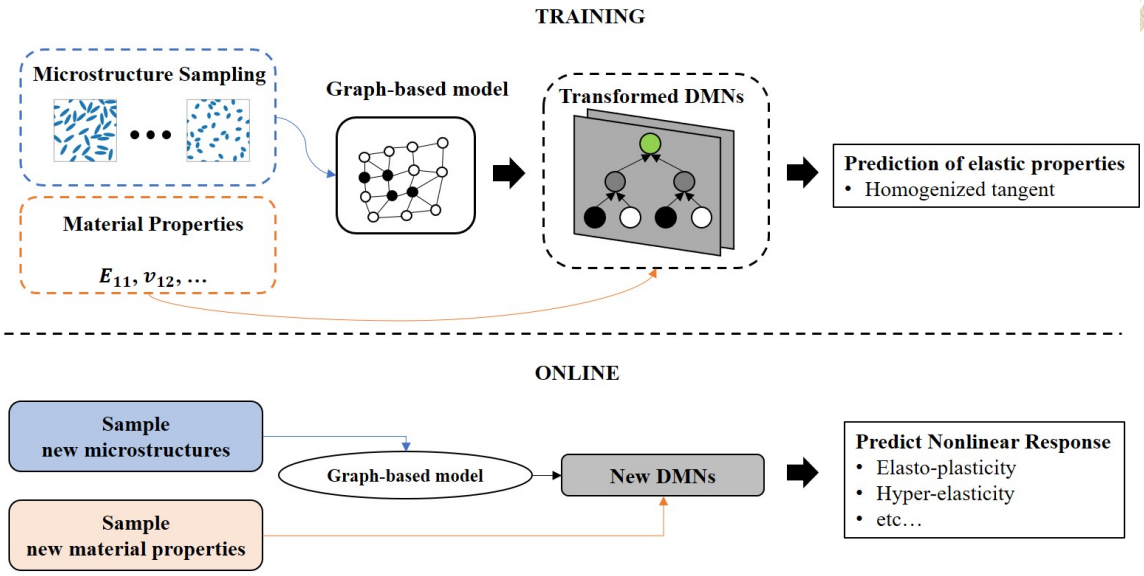



Figure 1.1: Illustration of the proposed framework for graph-based mechanistic learning on material behaviors of microstructures. During training, the graph-based model learns to generate DMNs able to predict elastic properties. During its online use, the model can generate new DMNs based on newly given microstructures. These given DMNs can in turn be used to model the nonlinear response of such microstructures.

In this work, we present a model (Fig. 1.1) able to handle dissimilar microstructures while taking advantage of the dimensionality-reduction capability of DMN.

Regarding the handling of different RVEs, we do so through a graph-based model. More specifically, we start from the discretized meshes of RVEs as approximations of their physical domain. Next, we convert these meshes into equivalent graph representations. A graph neural network then learns to compress these graph representations into meaningful geometric embeddings. As to the leveraging of the dimensionality-reduction capability of DMN, our model seeks to map these embeddings into individual DMNs. Those obtained DMNs can each make different predictions depending on the microstructures they are representing. For the purpose of illustration, the framework is applied to RVEs with circular and elliptical inclusions. The results of the training sets and tests on microstructures


doi:10.6342/NTU202301463

unseen during model fitting attest to the reliability of the approach. Overall, the results suggest that the model could potentially handle all existing families of microstructures, as long as the graph-neural network portion of its structure is expressive enough.

## 1.3 Structure of Thesis

The thesis is organized as follows. In Chapter 2, we review the literature relevant to our work, namely multiscale modeling from a mathematical point of view, as well as the use of machine learning in multiscale modeling. We additionally share our insights gained from the literature review. In Chapter 3, we introduce the main aspects of the deep material network, a surrogate model that is instrumental to our work. The gist of our work is presented in Chapter 4. In it, we explain the rationale behind our approach, the selected method for implementing it, as well as the architecture of our proposed model. Chapter 5 focuses on applications of the model. The first example considers a variety of RVEs with circular inclusions dispersed in their matrix. The second example deals with a case of RVEs with elliptical inclusions. Finally, in Chapter 6, we draw conclusions based on our results and envision further directions for future works.


doi:10.6342/NTU202301463

# Chapter 2 Literature Review

## 2.1 Multiscale Modeling

### 2.1.1 Composite Materials

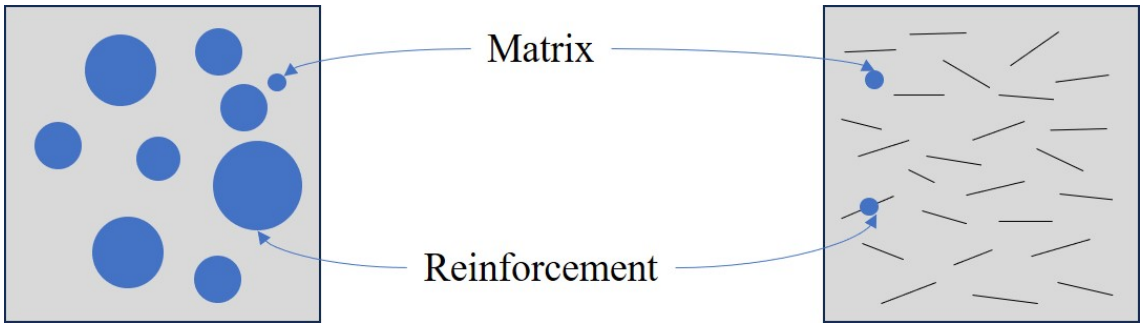



Figure 2.1: Examples of composite materials. A composite material is constituted of a matrix with reinforcements dispersed in that matrix.

This work mainly involves composite microstructures. It is therefore essential for us to review the definition of composite materials.

A *composite material*, or *composite* for short, is a type of material that is composed of at least two materials with different physical or chemical properties. The *filler* material is commonly referred to as the matrix, while the other materials play the role of reinforcements (Fig 2.1). Practically, the matrix acts as a medium that binds and holds the reinforcements together, protects them, and also provides load transfer. The reinforce-


doi:10.6342/NTU202301463

ments, embedded in the matrix, usually have high strength, stiffness and/or toughness. Together, the matrix and reinforcements combine to create the composite material with enhanced properties and performance characteristics. The matrix can be a polymer, ceramic, metal or carbon. The reinforcement can be fibers, fabric particles, or whiskers. Commonly used reinforcement materials include carbon fibers, glass fibers, silicon carbide fibers, aramid, etc. We may classify composites based on their constituent materials. Thus, we have ceramic matrix composites, metal matrix composites, polymer-based composites, etc. For instance, common polymer-based composite materials include glass-fiber reinforced plastic (GFRP) composites, carbon-fiber reinforced polymer (CFRP) composites and aramid-fiber reinforced polymer composites.

Composites have several advantages over traditional materials (e.g., steel, wood). They have a high strength-to-weight ratio. Steel, for instance, is five times as heavy as carbon fiber. Meanwhile, carbon fiber is five-times stronger than steel and twice as stiff. Composites are also durable. They do not suffer from corrosion like metals. Furthermore, they have high-impact strength, can be more easily moulded, designed and tailored for specific applications. For these reasons and their potential for increasing energy efficiency, composites have found widespread applications in virtually all industries. We may note the aerospace industry (e.g., engine blades, propellers, wings), the automotive industry (e.g., door panels, bumper, hood), sports and recreation (e.g., bicycles, hockey sticks, tennis racquets), architecture (e.g., interior/exterior finishes, insulation), marine applications.


doi:10.6342/NTU202301463

### 2.1.2 The Multiscale Modeling Process

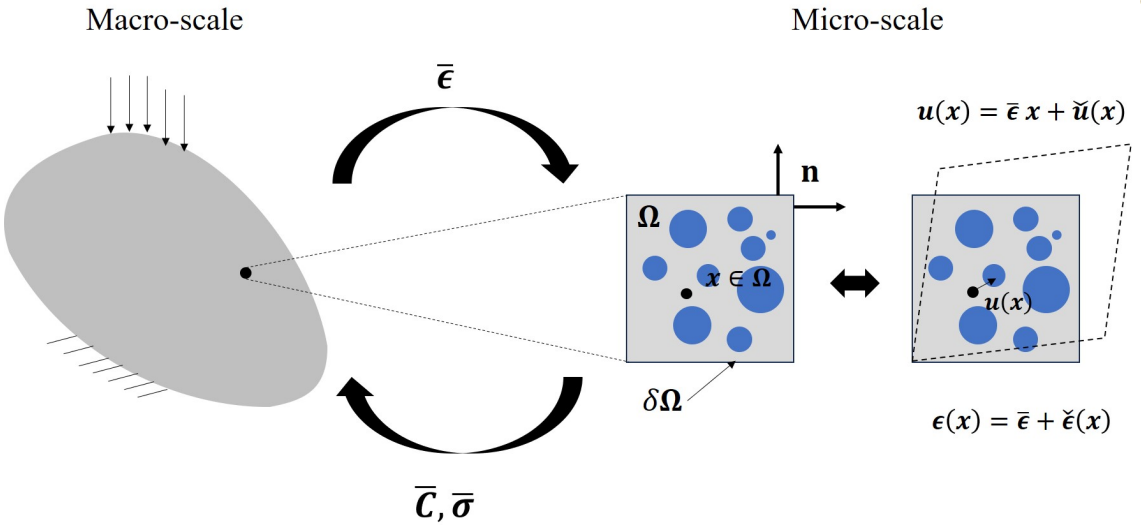
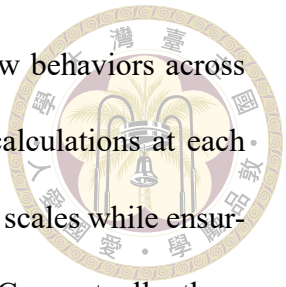



Figure 2.2: Illustration of the multiscale modeling process.

Composite materials have proved their benefits in different industries, and harbor as well hidden potentials yet to be discovered. In order to leverage all the advantages of composites, it is necessary to build a thorough understanding of their mechanics and behavior when subjected to different conditions. Hence, the ability to accurately simulate them becomes crucial. To do so, one may proceed to build a numerical model in which the details and heterogeneities across different length scales (e.g., microscale, mesoscale, macroscale) are all explicitly described. This task is however not practical, nor even feasible depending on the nature of the simulation. Supposing that the finite element method would be the technique used, one may end up with a discretization containing millions, even billions of elements. Accounting in addition for the time cost with nowadays computing power and the necessity of iterations shows the unrealistic nature of such an endeavor. Luckily, there is an approach that helps in mitigating many of those issues. This approach is none other than *multiscale modeling*.


doi:10.6342/NTU202301463



Multiscale modeling seeks to solve complex problems that show behaviors across multiple spatial and/or temporal scales. It does so by performing calculations at each scale using information from other scales, and by linking the different scales while ensuring the consistency (e.g., physical consistency) of the entire system. Conceptually, there exists two categories of multiscale approaches: the *sequential multiscale modeling* and the *concurrent multiscale modeling* [32]. In sequential multiscale modeling, the constitutive relations of large-scale models utilize details pre-computed by smaller-scale models. In concurrent multiscale modeling, the different scales are considered concurrently: the information needed at the larger scales are computed on-the-fly by the smaller scales. Thus an exchange of messages continuously happen between the scales. Sequential multiscale modeling is mostly used when the different scales are weakly linked and only a few parameters need to be shared between them. On the other hand, concurrent multiscale modeling is necessary when the different scales are strongly coupled, and thus need to continuously communicate.

### 2.1.3 Homogenization

For the purpose of composite material modeling, we are specifically concerned with concurrent multiscale modeling. It relies on an important tool: the *homogenization* technique. Through homogenization, we seek to compute the average properties of heterogeneous materials. Hence, one may average the properties of the *representative volume element* (RVE) of a composite material at the microscale, and then transfer that information to the macroscale. Research on homogenization can be dated back to 1887 with the introduction of the rule of mixtures [13]. Then, during the second half of the twentieth century, Hill [18, 19], Mori and Tanaka [34] published some of the most influential works


doi:10.6342/NTU202301463

on homogenization. The revolution of computers naturally gave birth to *computational homogenization*. Among the early works on computational homogenization, we find the publications of Suquet and others [33, 41].

Yvonnet [53] provides detailed information about computational homogenization in his book. Below we briefly present the computational homogenization problem for a RVE in the context of linear elasticity.

Let us consider a RVE with domain $\mathbf{\Omega}$, confined by the boundary $\delta\mathbf{\Omega}$ (Fig 2.2). The RVE is made up of $n$ constituent phases, the material properties of which are characterized by the fourth order elasticity tensors $\mathbb{C}^i$, $i = 1, 2, ..., n$. We consider this RVE to be subjected to a homogeneous, infinitesimal strain field $\bar{\boldsymbol{\epsilon}}$. The objective is to find the displacement field $\mathbf{u}(\mathbf{x})$ that satisfies the balance of linear momentum

$$\nabla \cdot \boldsymbol{\sigma}(\mathbf{u}(\mathbf{x})) = \mathbf{0} \quad \forall \mathbf{x} \in \mathbf{\Omega} \tag{2.1}$$

with the stress at each location $\mathbf{x}$ given by the local constitutive law

$$\boldsymbol{\sigma}(\mathbf{u}(\mathbf{x})) = \mathbb{C}^i(\mathbf{x}) : \boldsymbol{\epsilon}(\mathbf{u}(\mathbf{x})) \tag{2.2}$$

and the local strains within the RVE given by

$$\boldsymbol{\epsilon}(\mathbf{u}(\mathbf{x})) = \frac{1}{2}(\nabla \mathbf{u}(\mathbf{x}) + \nabla^T \mathbf{u}(\mathbf{x})) \tag{2.3}$$

The solution must additionally verify

$$\frac{1}{V_{RVE}} \int_{\Omega} \boldsymbol{\epsilon}(\mathbf{x})) \, d\Omega = \bar{\boldsymbol{\epsilon}} \tag{2.4}$$



doi:10.6342/NTU202301463

$V_{RVE}$ stands for the volume of the RVE. To solve the problem posed by equations 2.1, 2.2, 2.3, it is assumed that the local strains $\boldsymbol{\epsilon}$ are a superposition of the macroscopic field $\bar{\boldsymbol{\epsilon}}$ and some local strain fluctuations $\check{\boldsymbol{\epsilon}}$

$$\boldsymbol{\epsilon}(\mathbf{x}) = \bar{\boldsymbol{\epsilon}} + \check{\boldsymbol{\epsilon}}(\mathbf{x}) \tag{2.5}$$

Taking the average of equation 2.5 over the domain of the RVE, we obtain

$$\frac{1}{V_{RVE}} \int_{\Omega} \boldsymbol{\epsilon}(\mathbf{x})\, d\Omega = \bar{\boldsymbol{\epsilon}} + \frac{1}{V_{RVE}} \int_{\Omega} \check{\boldsymbol{\epsilon}}(\mathbf{x})\, d\Omega \tag{2.6}$$

Using the definition of infinitesimal strains, we may expand the previous equation into

$$\frac{1}{V_{RVE}} \int_{\Omega} \boldsymbol{\epsilon}(\mathbf{x})\, d\Omega = \bar{\boldsymbol{\epsilon}} + \frac{1}{2V_{RVE}} \int_{\Omega} (\nabla \check{\mathbf{u}}(\mathbf{x}) + \nabla^T \check{\mathbf{u}}(\mathbf{x}))\, d\Omega \tag{2.7}$$

Here, $\check{\mathbf{u}}(\mathbf{x})$ is a local displacement fluctuation such that

$$\mathbf{u}(\mathbf{x}) = \bar{\boldsymbol{\epsilon}}\, \mathbf{x} + \check{\mathbf{u}}(\mathbf{x}) \tag{2.8}$$

Applying Gauss's divergence theorem on equation 2.6, we obtain:

$$\frac{1}{V_{RVE}} \int_{\Omega} \boldsymbol{\epsilon}(\mathbf{x})\, d\Omega = \bar{\boldsymbol{\epsilon}} + \frac{1}{2V_{RVE}} \int_{\delta\Omega} (\check{\mathbf{u}}(\mathbf{x}) \otimes \mathbf{n} + \mathbf{n} \otimes \check{\mathbf{u}}(\mathbf{x}))\, dS_{RVE} \tag{2.9}$$

$S_{RVE}$ stands for the surface of the RVE.

Considering this final form, equation 2.4 will be satisfied if $\check{\mathbf{u}}(\mathbf{x}) = \mathbf{0}$, $\forall \mathbf{x} \in \delta\boldsymbol{\Omega}$; in other words if

$$\mathbf{u}(\mathbf{x}) = \bar{\boldsymbol{\epsilon}}\, \mathbf{x} \quad \forall \mathbf{x} \in \delta\boldsymbol{\Omega} \tag{2.10}$$

In this case, we are dealing with *kinematically uniform boundary conditions* (KUBC).


doi:10.6342/NTU202301463

Otherwise, equation 2.4 can also be satisfied if $\check{\mathbf{u}}(\mathbf{x})$ is periodic on the domain $\Omega$ such that

$$\mathbf{u}(\mathbf{x}) = \bar{\boldsymbol{\epsilon}}\,\mathbf{x} + \check{\mathbf{u}}(\mathbf{x}) \quad \forall \mathbf{x} \in \delta\boldsymbol{\Omega} \tag{2.11}$$

In that case, we are dealing with *periodic boundary conditions* (PBC). Moreover, there exists a third type of boundary conditions, which we have not reviewed here: the *stress uniform boundary conditions* (SUBC). The use of these three types of boundary conditions normally results in different estimations of the average properties computed over the RVE. Let us suppose that we wish to compute the homogenized elastic stiffness $\bar{\mathbb{C}}$ of the RVE satisfying $\bar{\boldsymbol{\sigma}} = \bar{\mathbb{C}} : \bar{\boldsymbol{\epsilon}}$, the results would be associated by the following inequalities

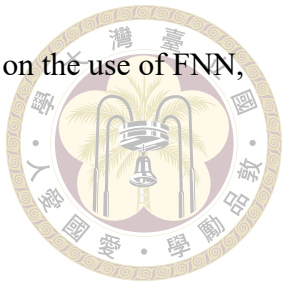
$$\bar{\mathbb{C}}_{(SUBC)} \leq \bar{\mathbb{C}}_{(PBC)} \leq \bar{\mathbb{C}}_{(KUBC)} \tag{2.12}$$

## 2.2 Machine Learning and Multiscale Modeling

Besides the traditional FEM or FFT-based multiscale modeling, machine learning (ML) techniques have been leveraged to carry out the same task. Several ML algorithms have been developed over the years, notably by researchers in the computer science field. We may list: linear regression, logistic regression, decision tree (DT), support vector machine (SVM), Naïve Bayes classifier, K-Nearest Neighbor(KNN) algorithm, feed-forward neural network (FNN), recurrent neural networks (RNN), convolutional neural networks (CNN), graph neural networks (GNN), deep reinforcement learning (DRL), etc. The list is endless. Due to recent improvements in computing power and the potential of ML methods to outperform traditional methods in speed, these techniques have gained traction in the computational mechanics community. A few papers [4, 5, 27] have reviewed their


doi:10.6342/NTU202301463

applications to multiscale modeling. Below we focus on our attention on the use of FNN, RNN, CNN, GNN, and additionally DMN in multiscale modeling.



### 2.2.1 Feed-forward Neural Network

A FNN consists of sequentially connected layers of neurons that share information between them. They do so through a set of *weights*, *biases* and *activation functions* that determine which and how information is carried out from a pre-defined *input* layer to a target *output* layer. Due to their simple, yet powerful, architecture, FNNs have been used for multiscale simulation. Lefik et al. [25] used FNNs to model the stress-strain behavior of composites. As another example, Yang et al. [51] combined knowledge of FNNs and insights from the theory of elastoplasticity to model the elastoplastic response of a composite beam.

### 2.2.2 Recurrent Neural Network

A more complex class of neural networks is the recurrent neural network. RNNs allow cycles in the passage of information. In other words, the outputs of a neuron can potentially affect future inputs to the same neuron. So-called *gates* control the mechanism of these cycles in popular RNN variants such as the Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU).

An obvious advantage of using RNNs is their suitability for predicting history-dependent behavior. Wu et al. [49] used a RNN-based model to predict the loading paths of a composite microstructure with an elasto-plastic matrix and hyperelastic fibers. Mozaffar et al. [35], in one of their examples, used an RNN to predict the stress response of a RVE with


doi:10.6342/NTU202301463

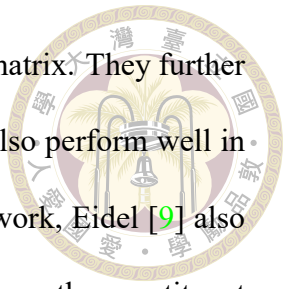
an AA6061 aluminum alloy matrix and rubber fillers. They additionally used the same model to predict its plastic energy and draw its yield surface. Ghavamian and Simone [14] modified a LSTM such that its gate mechanism became similar to the stress update process in standard nonlinear finite element analysis. The LSTM thus modified was used as a surrogate at the microscopic level in the context of the two-scale finite element ($FE^2$) simulation of a rectangular bar.

### 2.2.3 Convolutional Neural Network

Convolutional neural networks have gained popularity due to to their remarkable performance in various tasks related to computer vision and image processing. Such tasks include image classification, object detection, image segmentation, facial recognition, etc. The development of CNN started as far back as the 1980s. But, it was not until 2012, after the incredible performance of AlexNet [24] during the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), that the boom of research in CNN started. Since then various models have been developped. We may list GoogleNet [43], VGGNet [39], ResNet [17], Inception-ResNet [42], MobileNets [20].

Recently, CNNs have slowly drawn the attention of scientists in the computational mechanics field. Yang et al. [50] combined CNN and principal component analysis (PCA) to predict the nonlinear behavior of 100,000 binary composite microstructures. Within their framework, they used PCA to project the stress-strain curves of any composite into a low dimension. The goal of the CNN model was then to predict this reduced representation of the composite's nonlinear response. Rao and Liu [38] proposed a three-dimensional CNN able to predict the anisotropic elastic properties of RVEs such as their Young's and shear moduli, Poisson's ratio. They trained this model using a data set comprising 2000


doi:10.6342/NTU202301463



microstructures made of random spherical inclusions embedded in a matrix. They further fine-tuned this model via transfer learning and showed its ability to also perform well in predicting the properties of RVEs with ellipsoidal inclusions. In his work, Eidel [9] also used a 3D CNN architecture. He focused on two-phase microstructures, the constituent phases of which were distributed through random assignment and the use of Gaussian filters. The trained CNN predicted their homogenized elastic properties for three different types of boundary conditions: periodic boundary conditions, kinematically uniform boundary conditions, and stress uniform boundary conditions.

### 2.2.4 Graph Neural Network

Graph neural networks, as implied in the appellation, operate on graphs which are a type of unordered, non-Euclidean structured data. This is in contrast to CNNs, which are designed to analyze regular grid-like data such as images and time series. A graph is defined by nodes and edges linking them. GNNs are able to capture relational dependencies in graphs by leveraging message passing algorithms to propagate information through the nodes and edges of said graphs. This ability of GNNs, along with its greater flexibility compared to CNNs, has resulted in its utilization in a wide range of fields. These fields of application include: social network analysis, knowledge graph reasoning, recommendation systems, drug discovery, computer vision, natural language processing, logistics. We must also mention multiscale modeling.

Vlassis et al. [45] used GNNs to predict the hyperelastic energy functionals of polycrystals. To do so, they designed a hybrid network architecture in which graph convolutional networks [23] performed, on one hand, the unsupervised classification of the polycrystals into feature vectors. On another hand, a multi-layer perceptron performed the


doi:10.6342/NTU202301463

regression task of predicting energy functionals, given the feature vector and the second-order right Cauchy–Green deformation tensor of the polycrystals. In addition, they incorporated Sobolev training [8] while designing the cost function for the model to impose physical constraints on its outputs. Later, Vlassis and Sun [47] used a graph auto-encoder neural network architecture to compress plasticity graphs into vectors of internal variables. The aforementioned plasticity graphs were obtained by converting finite element meshes of RVEs and their corresponding elasto-plastic simulation results into graphs. The learned internal variables were then integrated into a component-based neural network plasticity model [46] to predict the elasto-plastic responses of the target RVEs with good fidelity. More recently, Jones et al. [22] presented a multi-level graph neural network architecture used for the tasks of heat conduction in polycrystals, stress evolution in polycrystalline materials subjected to plastic deformation, and stress evolution in viscoelastic composite materials. In the proposed model, GNNs separately perform message passing at different resolution levels of the target graphs. Meanwhile, specialized mappings enabled the upscaling and downscaling of information among levels. They reported that this hierarchical approach in their homogenization tasks surpassed the performance of popular graph pooling techniques such as DiffPool [52] and MinCutPool [3].

### 2.2.5 Deep Material Network

Last but not least, we overview the use of the deep material network in multiscale modeling. DMN, which was introduced in 2019 by Liu et al. [29], sets itself apart from the previously introduced neural network models on several aspects.

First, it was designed at its core for multiscale modeling. Popular neural network architectures (e.g., FNN, RNN, GNN) were designed to be as general as possible, thus


doi:10.6342/NTU202301463

rendering them applicable to different fields. Although they differ in their high-level formulation, they share at the lowest level *neurons* based on a linear polynomial function of the form $Wx + b$, in which $W, b$ are respectively a weight and bias to be optimized, while $x$ is a feature to be transformed. The powerful predictive powers of these neural networks stem from the stacking of several layers of neurons, the use of activation functions, the presence of mechanisms influencing the way neurons exchange information. In contrast, DMN has a building block based on two-phase laminate theory. It is assumed that a RVE, with phases stacked on top of each other, is subjected to a set of boundary conditions. The analytical form of the solution to determining the homogenized elastic tangents of this simple problem is then used as the building block for the network. The features to be transformed become the elastic tangents of the constituent phases, while the parameters to be optimized become, in part, the volume portions of theses phases. The parameters learned by DMN after training hold physical meanings and can be interpreted in relation to the geometry of the target microstructure.

The second, and main, distinguishing feature of DMN is its prediction capability. Although popular neural network architectures are applicable to a variety of fields, they currently face limiting factors in predicting material behavior during inference. One of such challenges is tied to the space of training data. For instance, a FNN or CNN fitted on linear elastic data can, broadly speaking, only be used to predict linear elastic properties. To predict complex material behavior (e.g., elasto-plasticity, damage, hyperelasticity), one needs to explicitly include such data at the training stage. DMN broke that barrier. It only needs linear elastic data during offline training. Yet, because of its ability to capture the physics of microstructures, it can be used to predict complex nonlinear behaviors during inference.


doi:10.6342/NTU202301463

Various publications have substantiated the close links between the fitted parameters of DMN and the topology of microstructures it is trained on. The first work on DMN [29] showed its capacity to implicitly learn (i.e., without being provided as inputs) the volume fractions of a diversity of two-dimensional RVEs. The simplest microstructure considered was a single-phase RVE. The other categories of RVEs studied had two constituent phases and were in three categories: matrix-inclusion, amorphous, and anisotropic. Liu and Wu [28], who explored the applicability of DMN to three-dimensional problems, reported similar observations. The trained 3D DMN was able to capture volume fractions when trained on a microstructure with spherical inclusions, and on a carbon fiber reinforced polymer (CFRP) composite as well. Gajek et al. [10] set to establish the basic mechanical principles of DMN and reported similar conclusions. The interested reader may additionally consult these works [37, 40, 48] for more proof.

However, this particularity of DMN is a double-edged sword. Its being able to capture the topology of complex microstructures with just a few degrees of freedom is astounding, to say the least. But, this characteristic also means that a DMN model can only be confidently used for the microstructures it was fitted on. Some researchers have taken notice of this limitation and have taken different approaches to increase the generalizability of DMN. Liu et al. [30] proposed a transfer learning approach. At an initial stage, a DMN is trained for a particle-reinforced 2D microstructure with very low fiber volume fraction. After fitting, they used some custom interpolation functions to extrapolate the parameters of the learned DMN for another RVE with higher fiber content. The thus newly-obtained parameters were then used as initialization for the training of the latter RVE. Using this procedure, they effectively accelerated the training of new DMNs. Huang et al. [21] built on the concept presented by Liu et al. [30] and proposed the so-called microstructure-guided


doi:10.6342/NTU202301463

DMN (MgDMN). In their approach, they trained *base* DMNs for microstructures at the extremities of a pre-defined parameter space. Afterwards, they used the *base* DMNs to interpolate new DMNs for intermediary microstructures within this parameter space. The previous two methods are *a posteriori* interpolation strategies in the sense that new DMNs are obtained through post-processing.

In contrast, Gajek et al. [11] proposed an *a priori* interpolation strategy. They modified the formulation of DMN such that orientation angles in the network are themselves functions of the fiber orientation tensor [1] of short-fiber reinforced composites. They trained this modified DMN for selected cases in the fiber orientation space, and then used it to interpolate properties for other microstructures within that space. A key advantage of this approach is that training is required only once. Nevertheless, it is not without its share of limitations. A microstructure generated based on a given fiber orientation tensor is not uniquely determined; there exists a randomness in that process. Hence, this model fails to account for the uncertainty arising from multiple microstructures realized based on the same fiber orientation tensor. Another limitation of this model is that its reformulation of DMN orientation angles was specifically designed for short-fiber reinforced composites. Hence, its extension to other material systems would not be straightforward.

## 2.3 Key Insights

The use of ML techniques for multiscale modeling is extensive. Each type of neural network architecture has its pros and cons. FNNs are the most general of all the neural networks surveyed here. However, they struggle at capturing history-dependency and complex relationships. RNNs address the issue of history-dependency, but they do so at


doi:10.6342/NTU202301463

the cost of a complex architecture. CNNs are good candidates for processing the geometry of microstructures, however they require grid-like data and might not be practical for predicting non-linearity. GNNs offer greater flexibility over CNNs as they can handle non-Euclidian structured data. Of all the sureveyed neural network architectures, DMN is the most promising one in multiscale modeling. Unlike the previously mentioned networks, it can be used for nonlinear prediction while being solely trained on linear elastic data. Moreover, it does so with significantly less parameters. However, its current limitation is its generalization to multiple microstructures. Given the flexibility of GNNs in handling non-Euclidian structured data (e.g., meshes) and the listed advantages of DMN, we have decided to combine the two into a single model GNN-DMN model. The architecture of this model will be detailed in Chapter 4.


doi:10.6342/NTU202301463



doi:10.6342/NTU202301463

# Chapter 3 Deep Material Network

Here, we adhere to the original formulation of DMN [29], slightly adjusted for the purpose of our work.

## 3.1 Notation

Given any linear elastic material, Hooke's law may be used to describe its response to applied deformations. The general equation for Hooke's law is:

$$\tilde{\sigma}_{ij} = \mathbb{C}_{ijkl}\tilde{\epsilon}_{kl} \tag{3.1}$$

Here, $\mathbb{C}_{ijkl}$ denotes the fourth-rank stiffness tensor. $\tilde{\sigma}_{ij}$ and $\tilde{\epsilon}_{kl}$ respectively denote the second-rank Cauchy stress tensor and second-rank infinitesimal strain tensor. Conversely, given the stress $\tilde{\sigma}_{kl}$, one may find the unknown strain $\tilde{\epsilon}_{ij}$ through the fourth-rank compliance tensor $\mathbb{D}_{ijkl}$:

$$\tilde{\epsilon}_{ij} = \mathbb{D}_{ijkl}\tilde{\sigma}_{kl} \tag{3.2}$$

In multi-linear algebra, fourth-rank and second-rank symmetric tensors may be reduced into matrices and vectors. For this purpose, the Voigt, Mandel, or other notation may be used. Standard books on finite elements for solid mechanics mostly use the Voigt


doi:10.6342/NTU202301463

notation. Although less popular in those manuals, the Mandel notation has the advantage of preserving the isometry of mapping, such that no weights need to be introduced when performing certain operations, as is the case with the Voigt notation. In Mandel notation, the 2D strain and stress may be written as:

$$\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2, \epsilon_3]^T \equiv [\tilde{\epsilon}_{11}, \tilde{\epsilon}_{22}, \sqrt{2}\tilde{\epsilon}_{12}]^T \tag{3.3}$$

$$\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \sigma_3]^T \equiv [\tilde{\sigma}_{11}, \tilde{\sigma}_{22}, \sqrt{2}\tilde{\sigma}_{12}]^T \tag{3.4}$$

Assuming plane-strain, the stiffness matrix $\boldsymbol{C}$, such that $\boldsymbol{\sigma} = \boldsymbol{C}\boldsymbol{\epsilon}$ is denoted by:

$$\boldsymbol{C} = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \equiv \begin{bmatrix} \mathbb{C}_{1111} & \mathbb{C}_{1122} & \sqrt{2}\mathbb{C}_{1112} \\ \mathbb{C}_{2211} & \mathbb{C}_{2222} & \sqrt{2}\mathbb{C}_{2212} \\ \sqrt{2}\mathbb{C}_{1211} & \sqrt{2}\mathbb{C}_{1222} & 2\mathbb{C}_{1212} \end{bmatrix} \tag{3.5}$$

while the compliance matrix $\boldsymbol{D}$, such that $\boldsymbol{\epsilon} = \boldsymbol{D}\boldsymbol{\sigma}$, is found by simply inverting the stiffness matrix:

$$\boldsymbol{D} = \begin{bmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{bmatrix} \equiv \boldsymbol{C}^{-1} \tag{3.6}$$


doi:10.6342/NTU202301463

## 3.2 Building Block

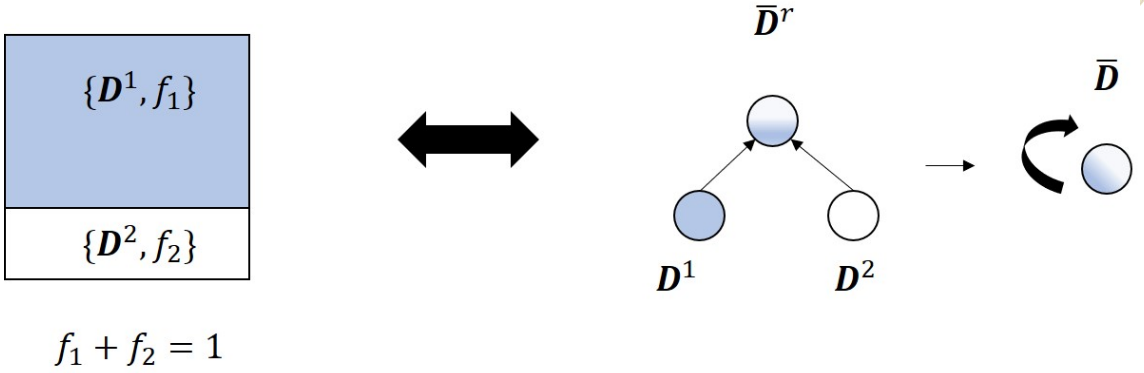



Figure 3.1: The building block of DMN, a two-phase RVE is represented as a single-layer binary tree.

Let us consider a two-phase laminate, as shown in Fig. 3.1. This laminate is assumed to be under plane-strain. The volume fractions of the constituent phases are respectively denoted by $f_1$ and $f_2 \equiv 1 - f_1$, and their stresses by $\boldsymbol{\sigma}^1$ and $\boldsymbol{\sigma}^2$. The averaging equations for strain and stress are respectively given by:

$$\boldsymbol{\epsilon} = f_1\boldsymbol{\epsilon}^1 + f_2\boldsymbol{\epsilon}^2 \tag{3.7}$$

$$\boldsymbol{\sigma} = f_1\boldsymbol{\sigma}^1 + f_2\boldsymbol{\sigma}^2 \tag{3.8}$$

Given the equilibrium conditions at the interface of the two-phase laminate:

$$\sigma_2^1 = \sigma_2^2, \quad \sigma_3^1 = \sigma_3^2 \tag{3.9}$$

and kinematic constraint:

$$\epsilon_1^1 = \epsilon_1^2 \tag{3.10}$$



doi:10.6342/NTU202301463

it is possible to analytically derive its homogenized tangent matrices. The homogenized stiffness matrix $\bar{\boldsymbol{C}}^r$ such that $\boldsymbol{\sigma} = \bar{\boldsymbol{C}}^r \boldsymbol{\epsilon}$ is:

$$\bar{\boldsymbol{C}}^r \equiv \mathbf{C}^2 - f_1 \Delta \mathbf{C} \boldsymbol{s}^1 \tag{3.11}$$

while the homogenized compliance matrix $\bar{\boldsymbol{D}}^r$ such that $\boldsymbol{\epsilon} = \bar{\boldsymbol{D}}^r \boldsymbol{\sigma}$ is:

$$\bar{\boldsymbol{D}}^r \equiv \boldsymbol{D}^2 - f_1 \Delta \boldsymbol{D} \boldsymbol{b}^1 \tag{3.12}$$

$\boldsymbol{s}^1$ and $\boldsymbol{b}^1$ respectively denote strain and stress localization matrices. Their definitions are provided in Appendix A.1. The detailed derivation for $\bar{\boldsymbol{C}}^r$ and $\bar{\boldsymbol{D}}^r$ are given in Appendix A.2.

After homogenizing the tangents as derived analytically, the authors of DMN added a rotation step to render its building block more general. Given $\bar{\boldsymbol{D}}^r$, the resulting $\bar{\boldsymbol{D}}$ after rotation by an angle $\theta$ may be written as:

$$\bar{\boldsymbol{D}} = \boldsymbol{R}(-\theta) \bar{\boldsymbol{D}}^r \boldsymbol{R}(\theta) \tag{3.13}$$

with the rotation matrix $\boldsymbol{R}(\theta)$ in Mandel notation given by:

$$\boldsymbol{R}(\theta) := \begin{bmatrix} \cos^2\theta & \sin^2\theta & \sqrt{2}\sin\theta\cos\theta \\ \sin^2\theta & \cos^2\theta & -\sqrt{2}\sin\theta\cos\theta \\ -\sqrt{2}\sin\theta\cos\theta & \sqrt{2}\sin\theta\cos\theta & \cos^2\theta - \sin^2\theta \end{bmatrix} \tag{3.14}$$

This concludes the formulation of the building block of DMN.


doi:10.6342/NTU202301463

## 3.3 Network Topology

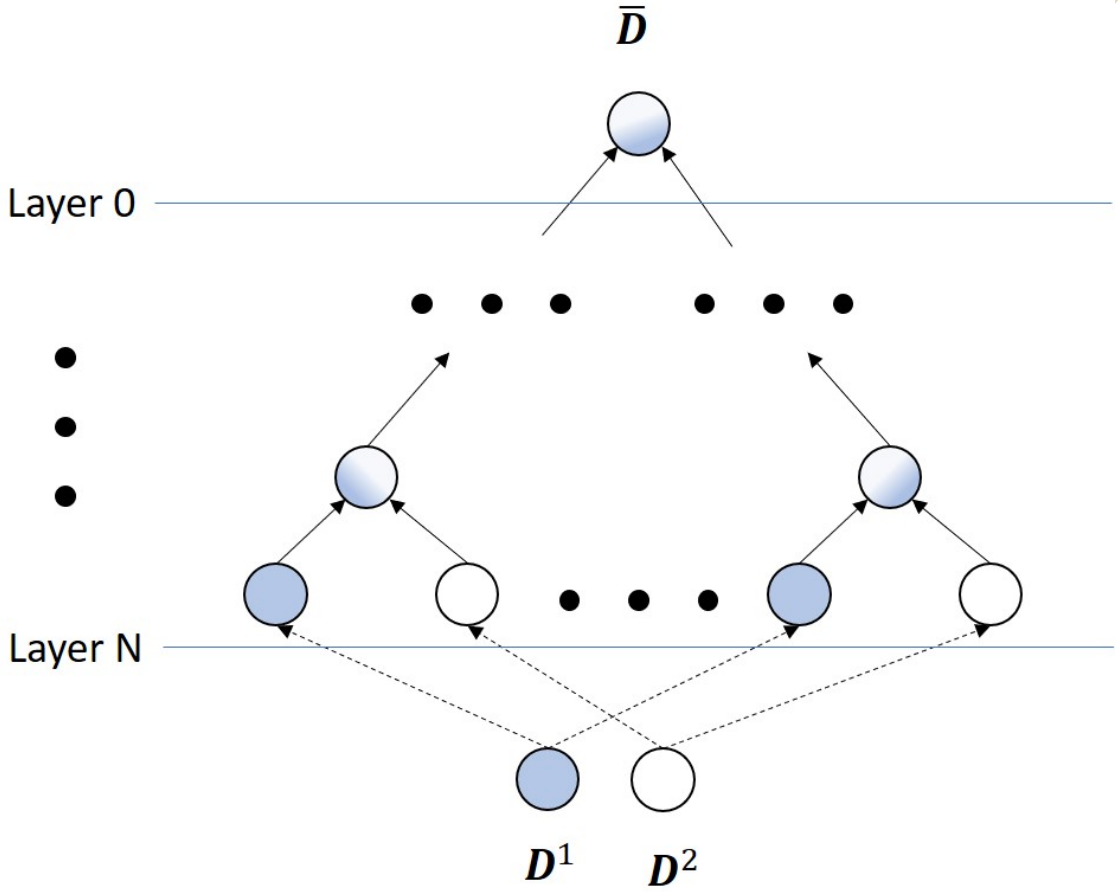



Figure 3.2: The topology of DMN.

The building block (Fig. 3.1) previously described can be thought of as a *complete* binary tree of depth $N = 1$ (Fig. 3.2). The *root* node of the tree, located at *level* or *layer* 0, stores the homogenized tangent $\bar{\boldsymbol{D}}$. Meanwhile, the leaf nodes (located at layer N) contain the material properties $\boldsymbol{D}^1$ and $\boldsymbol{D}^2$. To obtain a network with more complexity, several building blocks are stack on top of each other. A $N$-layer DMN is composed of a total of $2^{N+1} - 1$ nodes, each level $h$ of the tree having $2^h$ nodes.

As introduced, the building block requires, besides material properties, volume fractions and a rotation angle. In a $N$-layer DMN, angles $\theta_{h=0,1,\ldots,N}^{k=1,2,\ldots,2^h}$ are assigned at every


doi:10.6342/NTU202301463

node the network. Regarding volume fractions, they are indirectly calculated. Activations $z_N^{k=1,2,\ldots,2^N}$ are assigned at the bottom layer $N$. Abusing notation, these activations $z_N^k$ are then passed through an activation function to obtain weights $w_N^k$. In their work, Liu et al. [29] selected the Rectified Linear Unit (ReLU) activation function for that purpose, and additionally utilized *node deletion* techniques to handle nodes with null weights. Within our framework, we require DMN to conserve the totality of its nodes. Hence, we opt instead for the Softplus activation function such that

$$w_N^k = \log(1 + \exp(z_N^k)) \tag{3.15}$$

This approach guarantees the positivity of all weights at the bottom layer. Weights of nodes in upper layers of the network are then computed through the relationship

$$w_h^k = w_{h+1}^{2k-1} + w_{h+1}^{2k} \tag{3.16}$$

implying that the weight of each node is the summation of the weight of its *children* nodes. Conversely, the volume fraction of each node is computed as the ratio of its weight to that of its *parent* node

$$f_{h+1}^{2k-1} = w_{h+1}^{2k-1} / w_h^k \tag{3.17}$$

As to the material properties, they are initially assigned at layer $N$ (Fig. 3.2), and flow through the network as follows. The tangent at each node is rotated by the corresponding angle $\theta_h^k$ (Eq. 3.14). The rotated tangent of each node, together with its volume fraction $f_h^k$, are then used as values necessary to compute the homogenized tangent (Eq. 3.11 and Eq. 3.12) that will become the input of its parent node. The material properties are thus progressively transformed until the root of the tree is reached. The final output $\bar{\boldsymbol{D}}$ of DMN


doi:10.6342/NTU202301463

may then be described by the following function $\mathcal{F}$ such that

$$\bar{\boldsymbol{D}} = \mathcal{F}(\boldsymbol{D}^1, \boldsymbol{D}^2 |\, z_N^{k=1,2,\ldots,2^N}, \theta_{h=0,1,\ldots,N}^{k=1,2,\ldots,2^h}) \tag{3.18}$$

## 3.4 Data Generation

To compute the homogenized tangent matrix $\bar{\boldsymbol{D}}$ for a particular RVE, we carry out a set of three tests: two uniaxial tests and one in-plane shear test. For each test, an (infinitesimal) macroscopic strain is applied in the target direction while the remaining directions are left unconstrained. This procedure can be carried out with any finite-element tool. In our case, we had used the RVE package [31] available in LS-DYNA®.

For the first test, we constrain the RVE with $\tilde{\epsilon}_{11} = \delta$, where $\delta$ is a non-zero small number (e.g., $\delta = 10^{-12}$). The other strain components are left unconstrained as mentioned above. After the completion of the FE simulation, we obtain the stress and strain response of the RVE as:

$$\boldsymbol{\epsilon} = [\tilde{\epsilon}_{11}, \tilde{\epsilon}_{22}, \tilde{\epsilon}_{12}]^T, \boldsymbol{\sigma} = [\tilde{\sigma}_{11}, 0, 0]^T$$

Given these, we compute three components of the fourth-rank compliance tensor $\mathbb{D}_{ijkl}$:

$$\mathbb{D}_{1111} = \tilde{\epsilon}_{11}/\tilde{\sigma}_{11}, \mathbb{D}_{2211} = \tilde{\epsilon}_{22}/\tilde{\sigma}_{11}, \mathbb{D}_{1211} = \tilde{\epsilon}_{12}/\tilde{\sigma}_{11} \tag{3.19}$$

For the second uniaxial test, we apply a macroscopic uniaxial loading $\tilde{\epsilon}_{22} = \delta$, which allows us to obtain:

$$\boldsymbol{\epsilon} = [\tilde{\epsilon}_{11}, \tilde{\epsilon}_{22}, \tilde{\epsilon}_{12}]^T, \boldsymbol{\sigma} = [0, \tilde{\sigma}_{22}, 0]^T$$



doi:10.6342/NTU202301463

from which we compute

$$\mathbb{D}_{1122} = \tilde{\epsilon}_{11}/\tilde{\sigma}_{22}, \mathbb{D}_{2222} = \tilde{\epsilon}_{22}/\tilde{\sigma}_{22}, \mathbb{D}_{1222} = \tilde{\epsilon}_{12}/\tilde{\sigma}_{22} \tag{3.20}$$

Finally, we apply a macroscopic shear loading $\tilde{\epsilon}_{12} = \delta$, which allows us to obtain:

$$\boldsymbol{\epsilon} = [\tilde{\epsilon}_{11}, \tilde{\epsilon}_{22}, \tilde{\epsilon}_{12}]^T, \boldsymbol{\sigma} = [0, 0, \tilde{\sigma}_{12}]^T$$

from which we compute

$$\mathbb{D}_{1112} = \tilde{\epsilon}_{11}/\tilde{\sigma}_{12}, \mathbb{D}_{2212} = \tilde{\epsilon}_{22}/\tilde{\sigma}_{12}, \mathbb{D}_{1212} = \tilde{\epsilon}_{12}/\tilde{\sigma}_{12} \tag{3.21}$$

In our presentation of the theory of DMN, we had mentioned that the 2D stiffness matrix $\bar{\boldsymbol{C}}$, such that $\boldsymbol{\sigma} = \bar{\boldsymbol{C}}\boldsymbol{\epsilon}$, is given in Mandel notation by

$$\bar{\boldsymbol{C}} = \begin{bmatrix} \mathbb{C}_{1111} & \mathbb{C}_{1122} & \sqrt{2}\mathbb{C}_{1112} \\ \mathbb{C}_{2211} & \mathbb{C}_{2222} & \sqrt{2}\mathbb{C}_{2212} \\ \sqrt{2}\mathbb{C}_{1211} & \sqrt{2}\mathbb{C}_{1222} & 2\mathbb{C}_{1212} \end{bmatrix} \tag{3.22}$$

It can inversely be shown that the 2D compliance matrix $\bar{\boldsymbol{D}}$, such that $\boldsymbol{\epsilon} = \bar{\boldsymbol{D}}\boldsymbol{\sigma}$ is given by

$$\bar{\boldsymbol{D}} = \bar{\boldsymbol{C}}^{-1} \equiv \begin{bmatrix} \mathbb{D}_{1111} & \mathbb{D}_{1122} & \frac{\mathbb{D}_{1112}}{\sqrt{2}} \\ \mathbb{D}_{2211} & \mathbb{D}_{2222} & \frac{\mathbb{D}_{2212}}{\sqrt{2}} \\ \frac{\mathbb{D}_{1211}}{\sqrt{2}} & \frac{\mathbb{D}_{1222}}{\sqrt{2}} & \frac{\mathbb{D}_{1212}}{2} \end{bmatrix} \tag{3.23}$$

Using equations 3.19, 3.20, 3.21, and 3.23, we easily calculate all the components of the homogenized compliance matrix $\bar{\boldsymbol{D}}$.


doi:10.6342/NTU202301463

## 3.5 Model Training

The parameters to be fitted in DMN are the activations $z_N^k$ and rotation angles $\theta_h^k$ (Eq. 3.18). At initialization, they are assigned random values. Following Liu et al. [29], they are sampled $z_N^k$ and $\theta_h^k$ as follows

$$z_N^k \sim U(0.2, 0.8), \theta_h^k \sim U(-\pi/2, \pi/2) \tag{3.24}$$

where $U$ stands for uniform distribution. We can expect the output $\bar{\boldsymbol{D}}$ of the network based on such an initialization to be sub-optimal. Hence, we may use a gradient descent algorithm to tune $z_N^k$ and $\theta_h^k$ into optimal values. In gradient descent, the objective is to minimize a cost function designed to penalize departures of given predictions from specific targets. The following cost function $J$ is defined:

$$J(\mathcal{F}) = \frac{1}{N_s} \sum_{i}^{N_s} \frac{\|\bar{\boldsymbol{D}}_i^{pred} - \bar{\boldsymbol{D}}_i^{DNS}\|_F^2}{\|\bar{\boldsymbol{D}}_i^{DNS}\|_F^2} + \lambda L(w_N^k) \tag{3.25}$$

in which $\bar{\boldsymbol{D}}_i^{pred}$ and $\bar{\boldsymbol{D}}_i^{DNS}$ respectively denote the tangents predicted by DMN and those obtained via direct numerical simulation (DNS). $N_s$ represents the number of samples in the batch of data being evaluated. $\|\cdot\|_F$ denotes the Frobenius norm. The Frobenius norm $\|\boldsymbol{A}\|_F$ of a matrix $\boldsymbol{A}$ is defined as

$$\|\boldsymbol{A}\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2} \tag{3.26}$$



doi:10.6342/NTU202301463

The second term $\lambda L(w_N^k)$ in the cost function (Eq. 3.25) is a regularization term. $\lambda$ is a hyperparameter to be selected. $L(w_N^k)$ is defined as

$$L(w_N^k) = \left( \sum_k^{2^N} w_N^k - \alpha \right)^2 \tag{3.27}$$

in which $\alpha$ is a scalar that controls the magnitudes of the fitted weights, such that $\sum_k^{2^N} w_N^k = \alpha$. As in [28–30], we set it equal to $2^{N-2}$. Considering Eq. 3.15, Eq. 3.27 can further be written as

$$L(w_N^k) = \left( \sum_k^{2^N} \log(1 + \exp(z_N^j)) - \alpha \right)^2 \tag{3.28}$$

Beside the cost function $J$ which is used to optimize the network, an additional error function is defined as

$$\text{average error} = \frac{1}{N_s} \sum_i^{N_s} \frac{\|\bar{\boldsymbol{D}}_i^{pred} - \bar{\boldsymbol{D}}_i^{DNS}\|_F}{\|\bar{\boldsymbol{D}}_i^{DNS}\|_F} \tag{3.29}$$

It is used to track the accuracy of the predicted tangents throughout the training process.



doi:10.6342/NTU202301463

## 3.6 Prerequisite for Nonlinear Prediction

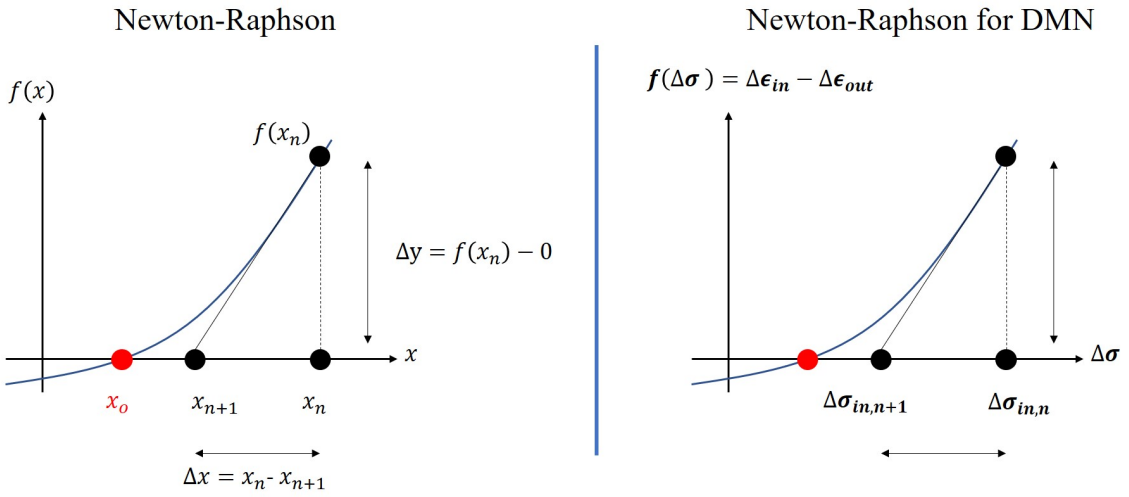



Figure 3.3: The Newton-Raphson method.

### 3.6.1 The Newton-Raphson Method

Let us suppose that we have a function $f(x)$, as shown in Fig 3.3, for which we want to find the root $x_o$ such that $f(x_o) = 0$. Using Newton's method, we start by arbitrarily guessing a value $x_n$. By plugging $x_n$ into the expression for $f(x)$, we easily obtain $f(x_n)$.

Our next step is to get closer to the root by estimating $x_{n+1} \equiv x_n - \Delta x$. To do so, we need to compute the value of the step $\Delta x$. The value of the slope of $f(x)$ at $x_n$ may be written as:

$$f'(x_n) \approx \frac{\Delta y}{\Delta x} = \frac{f(x_n) - 0}{\Delta x}$$

which gives:

$$\Delta x = \frac{f(x_n)}{f'(x_n)}$$



doi:10.6342/NTU202301463

$x_{n+1}$ can therefore be computed as:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

At $x_{n+1}$, we are closer to the root, but not quite there yet. We can then compute $x_{n+2}$ by starting from $x_{n+1}$, and so on, until we estimate that we have approached the root enough by a given tolerance.

### 3.6.2 Newton-Raphson applied to DMN

In the context of DMN, we have a tree representing an RVE to which we apply an (external) incremental strain $\Delta\bar{\boldsymbol{\epsilon}}$ as boundary condition. Our goal is to find the (internal) incremental stress $\Delta\boldsymbol{\sigma}_0$ due to the response of the RVE due to this applied strain. The (internal) incremental strain $\Delta\boldsymbol{\epsilon}_0 \equiv \boldsymbol{D}\Delta\boldsymbol{\sigma}_0 + \delta\boldsymbol{\epsilon}_0$ of the RVE should ideally be equal to the external strain $\Delta\bar{\boldsymbol{\epsilon}}$.

| | |
|---|---|
| x $\rightarrow$ | $\Delta\boldsymbol{\sigma}_0$ |
| f(x) $\rightarrow$ | $\Delta\boldsymbol{\epsilon}_0 - \Delta\bar{\boldsymbol{\epsilon}} \equiv (\boldsymbol{D}_0\Delta\boldsymbol{\sigma}_0 + \delta\boldsymbol{\epsilon}_0) - \Delta\bar{\boldsymbol{\epsilon}}$ |
| f'(x) $\rightarrow$ | $\boldsymbol{D}_0$ |

Table 3.1: Newton-Raphson variables in the context of DMN

We may draw a correspondence between variables in DMN with those in the formula for the Newton-Raphson procedure, as shown in Table 3.1. The formula for updating the incremental stress can thence be deduced as

$$\Delta\boldsymbol{\sigma}_0 \leftarrow \Delta\boldsymbol{\sigma}_0 - \frac{\Delta\boldsymbol{\epsilon}_0 - \Delta\bar{\boldsymbol{\epsilon}}}{\boldsymbol{D}_0} \tag{3.30}$$



doi:10.6342/NTU202301463

## 3.7 Nonlinear Prediction

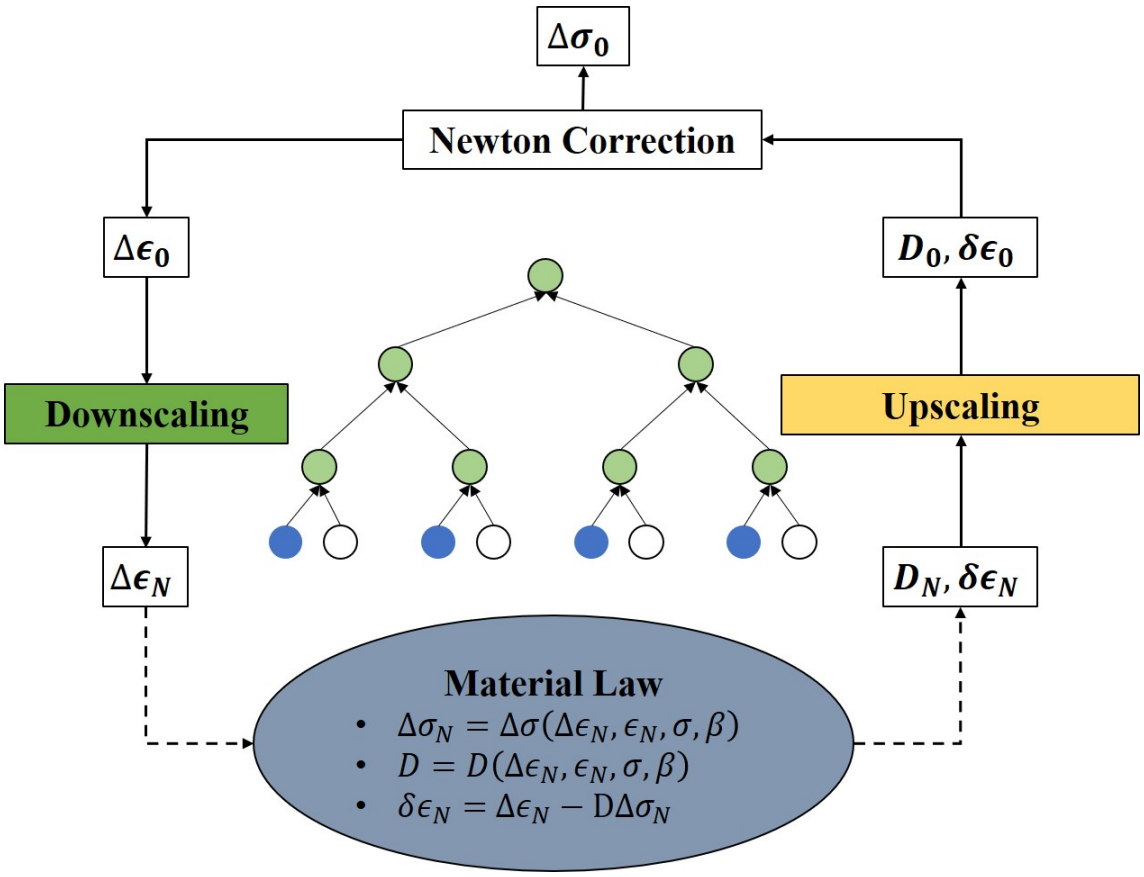



Figure 3.4: The data workflow during nonlinear prediction.

Fig. 3.4 illustrates the workflow for nonlinear prediction using DMN. The corresponding details are provided in Algorithm 1 within Appendix B.1.

Before starting the Newton-Raphson procedure, we initialize the incremental stresses $\Delta\boldsymbol{\sigma}_h^k$ and incremental strains $\Delta\boldsymbol{\epsilon}_h^k$ to $\mathbf{0}$ at all layers of the DMN tree. For the bottom layer, we additionally initialize the residual strains $\delta\boldsymbol{\epsilon}_N^k$ to $\mathbf{0}$, and the compliance matrices $\boldsymbol{D}_N^k$ to values provided by corresponding linear elastic laws.


doi:10.6342/NTU202301463

### 3.7.1 Upscaling

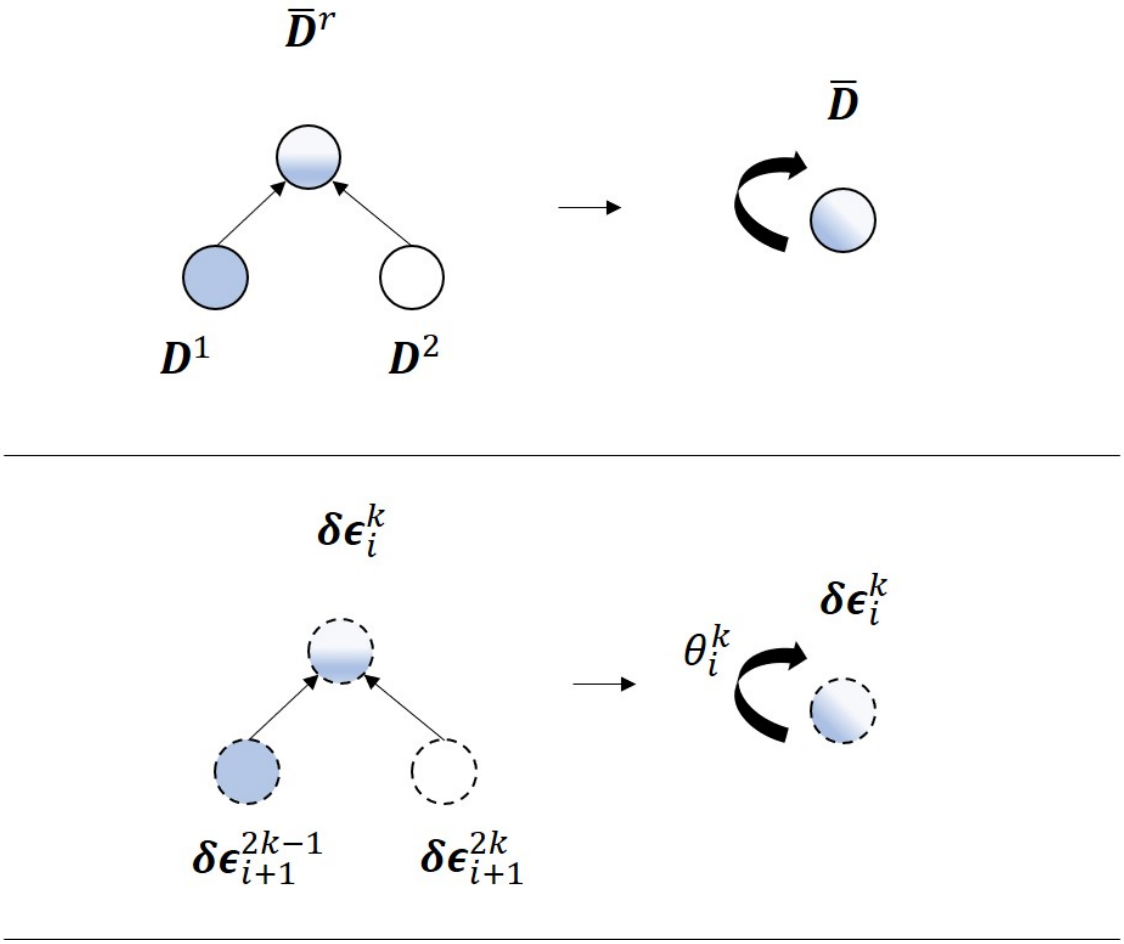



Figure 3.5: The homogenizaton step.

With residual strains and tangents known at the bottom nodes, they are homogenized until the top node of the DMN tree is reached through their respective homogenization (Algorithm 3) and rotation (Algorithm 6) functions illustrated in Fig. 3.5. In the end, we obtain new values for $\boldsymbol{D}_0$ and $\delta\boldsymbol{\epsilon}_0$ at the root node.


doi:10.6342/NTU202301463

### 3.7.2 Application of Boundary Conditions

Now that the top node is reached, the incremental stress $\Delta\boldsymbol{\sigma}_0$ may be corrected using the expression previously derived via Newton-Raphson. The expression in Eq. 3.30 is evaluated only for the components at which the macroscopic strain constraint is applied. With the incremental stress at the top node now known, the incremental strain there may be computed through the relationship $\Delta\boldsymbol{\epsilon}_0 = \boldsymbol{D}_0\Delta\boldsymbol{\sigma}_0 + \delta\boldsymbol{\epsilon}_0$.

### 3.7.3 Downscaling

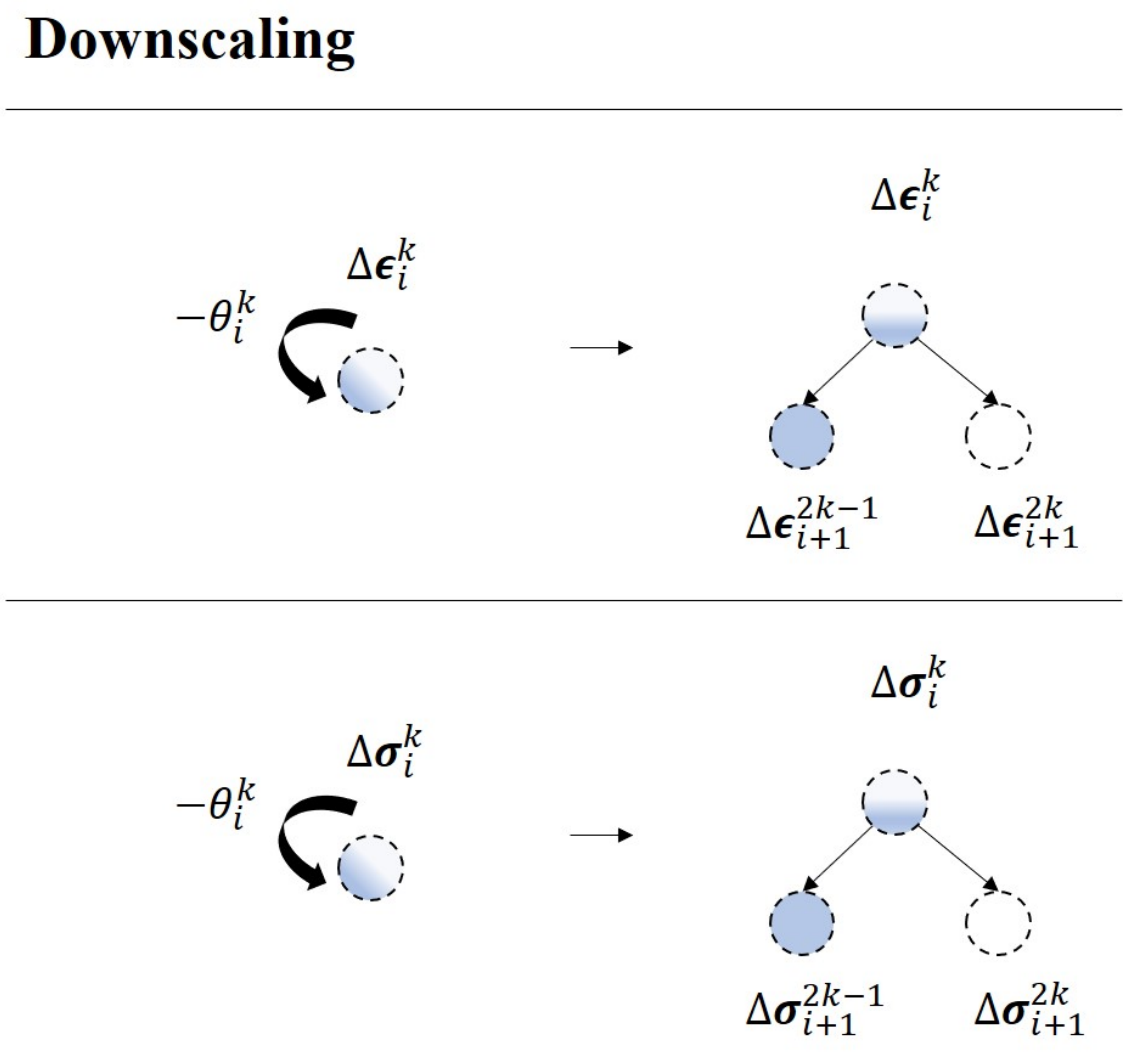



Figure 3.6: The de-homogenization step.



doi:10.6342/NTU202301463

Next, we recursively distribute the incremental strain $\Delta\boldsymbol{\epsilon}_0$ from the top node until the bottom nodes of the DMN tree are reached (Fig. 3.6) by using the strain concentration matrices and rotation angles (Algorithm 4). The same is done with the incremental stress $\Delta\boldsymbol{\sigma}_0$ using the stress concentration matrices and rotation angles (Algorithm 5).

### 3.7.4 Evaluation of constitutive laws at bottom layer

Once the bottom layer N is reached, the material laws at those nodes are used to update the tangents and residual strains.

$$\Delta\boldsymbol{\sigma}_N = \Delta\boldsymbol{\sigma}(\Delta\boldsymbol{\epsilon}_N, \epsilon_{\mathbf{N}}, \boldsymbol{\sigma}_{\mathbf{N}}, \boldsymbol{\beta}_{\mathbf{N}}) \tag{3.31}$$

$$\boldsymbol{D}_N = \boldsymbol{D}(\Delta\boldsymbol{\epsilon}_N, \epsilon_{\mathbf{N}}, \boldsymbol{\sigma}_{\mathbf{N}}, \boldsymbol{\beta}_{\mathbf{N}}) \tag{3.32}$$

$\boldsymbol{\beta}_N$ represents history variables at the bottom layer. In the case of elasto-plasticity, the main history variable is the accumulated plastic strain. The new residual strains are given by:

$$\delta\boldsymbol{\epsilon}_N = \Delta\boldsymbol{\epsilon}_N - \boldsymbol{D}_N\Delta\boldsymbol{\sigma}_N \tag{3.33}$$

### 3.7.5 Convergence Check

At the bottom layer, we compare the newly obtained incremental strains $\Delta\boldsymbol{\epsilon}_N$ with those from the previous iteration. If the change is small, we may move to the next loading step. Otherwise, we use the new residual strains and compliance matrices obtained by evaluating the constitutive law at the bottom layer to proceed to the ”Upscaling” step.


doi:10.6342/NTU202301463

This entire procedure is detailed in Algorithm 1 within Appendix B.1.

## 3.8 Example

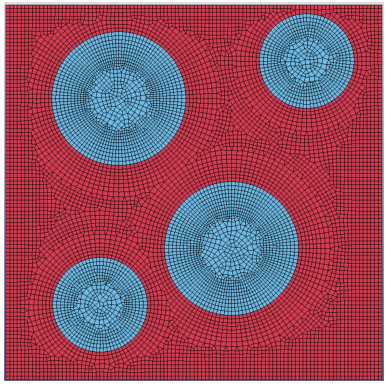


Figure 3.7: RVE with two constituent phases. The matrix is shown in red, while the inclusions are shown in blue. The RVE is meshed into 10,419 elements.

To illustrate the training and use of DMN, we consider a RVE constituted of a matrix surrounding circular inclusions (Fig. 3.7). The volume fractions of the matrix and inclusions are respectively 70.12% and 29.88%. This RVE is meshed into 10,419 elements using a mixed discretization scheme (quadrilateral and triangular elements).

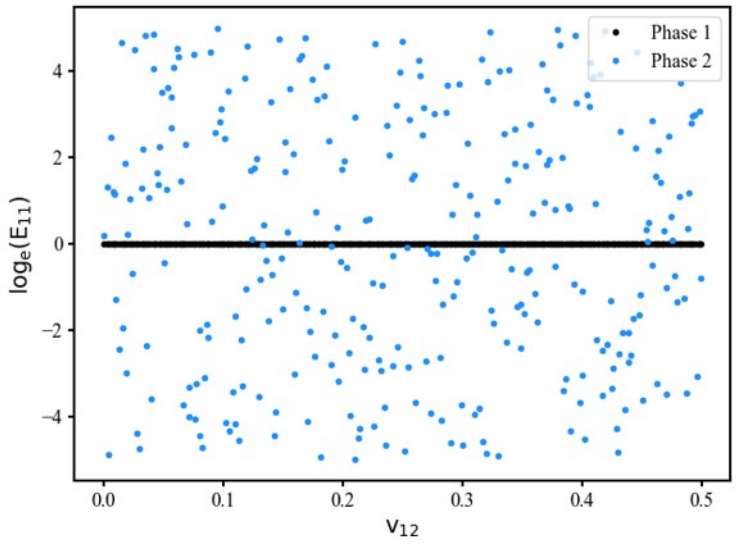



Figure 3.8: Sampling of elastic properties for the constituent phases of the RVEs. The Young's modulus is held constant in phase 1, while it varies in phase 2.



doi:10.6342/NTU202301463

In order to train the DMN, we need to sample material properties for the constituent phases of the RVE. We assume the plane-strain condition and that both phases are linear isotropic. We make use of Latin hypercube sampling (LHS) to generate the sampling space (Fig 3.8) of material properties for the RVE. More specifically, the sampling space consists of 300 data points for each phase. The Young's modulus is held equal to unity ($e^0$) for phase 1 (matrix), while it varies between $e^{-5}$ and $e^5$ for phase 2 (inclusions). In other words, the elastic modulus for the matrix can be approximately 150 times as high or as small as that of the inclusions. As for the Poisson's ratio, it varies between 0 and 0.5 for both phases. From these data points, we easily construct the input data to DMN, namely the compliance matrices $\boldsymbol{D}^1$ and $\boldsymbol{D}^2$. As to the target values, we procedure described in Chapter 3.4 to obtain the homogenized tangents $\bar{\boldsymbol{D}}$. From the 300 pieces of data generated, 200 were used for training, while the remaining 100 were used for validation.

### 3.8.1 Training

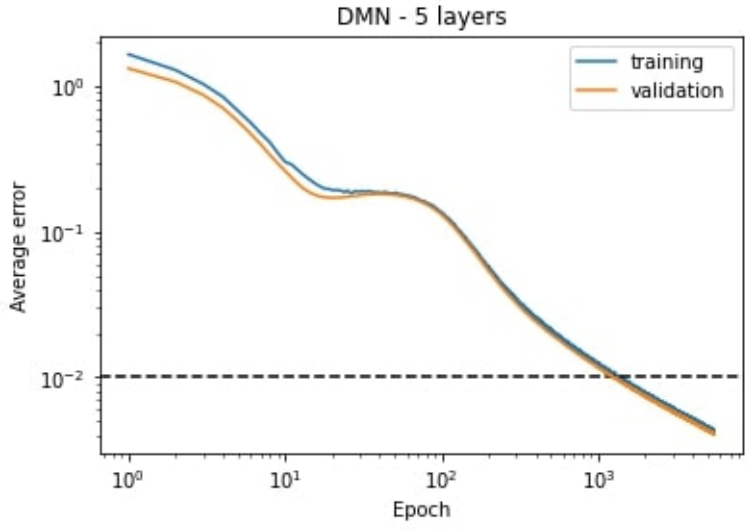
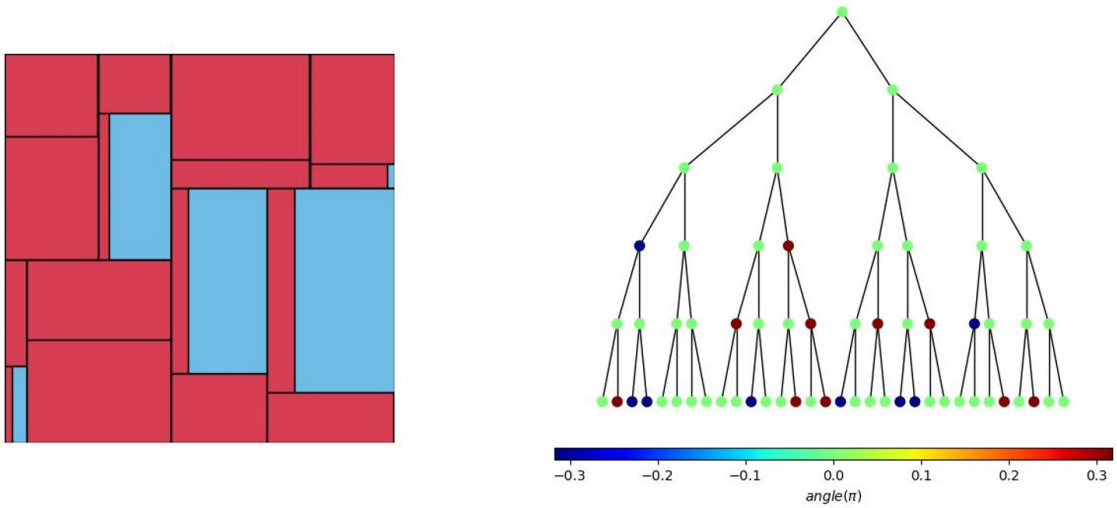



Figure 3.9: The 5-layer DMN is trained for over 5000 epochs. At the end of the fitting process, the average errors for the training and validation sets are below 1%.


doi:10.6342/NTU202301463

We train a 5-layer DMN for over 5000 epochs. By looking at the training and validation curves, we notice trends similar to results reported by Liu et al. [29]. No overfitting is observed; the training and validation curves are close to each other. Moreover, the training and validation errors decreased to less than 1%. To be more specific, the average errors for the training and validation sets are respectively 0.43% and 0.41%.

### 3.8.2 Learned Physics



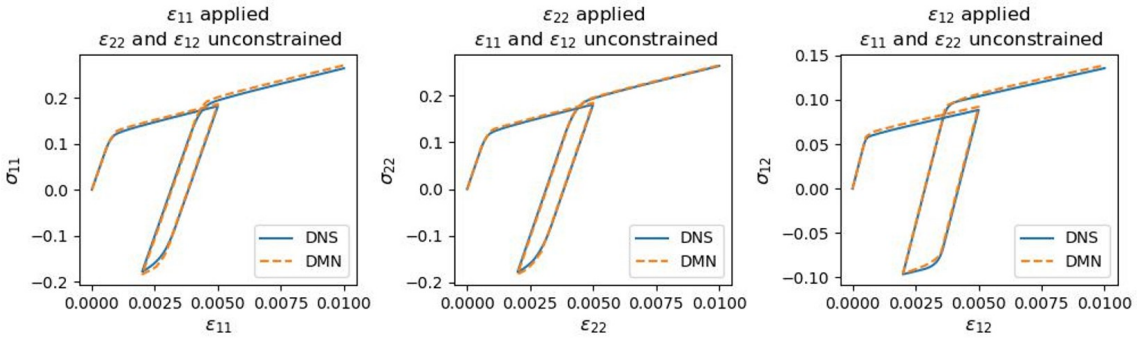

**(a)** **(b)**

Figure 3.10: Visualization of the parameters learned by DMN, respectively (a) its weights and (b) rotation angles.

Then, we inspect the parameters learned by DMN. Based on the learned weights of DMN, we compute the volume fractions throughout its tree and visualize them using a treemap (Fig. 3.10a). As to the rotation angles in the network, we directly visualize them on the DMN tree (Fig. 3.10b). The trained DMN learns an inclusion volume fraction of 30.07%, which is very close to the truth (29.88%).


doi:10.6342/NTU202301463

### 3.8.3 Nonlinear Prediction

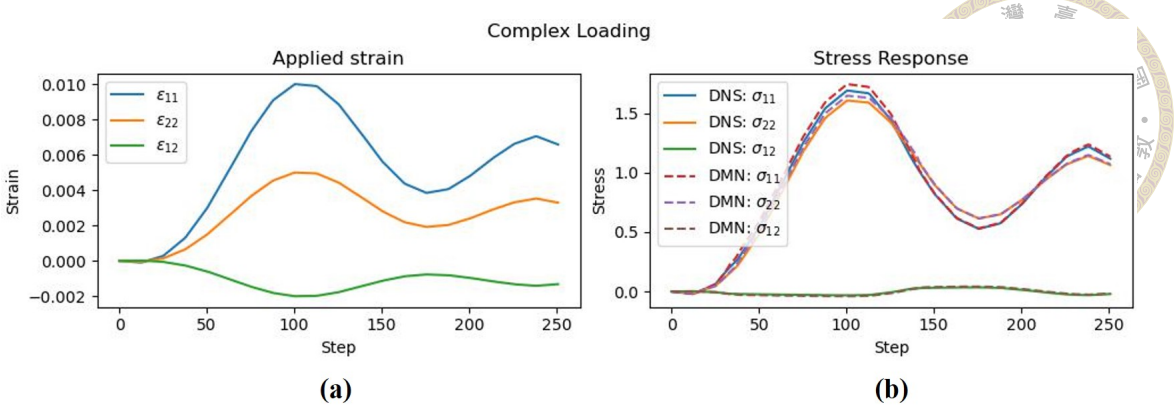



**(a)** **(b)** **(c)**

Figure 3.11: Nonlinear prediction by DMN for three simple tests: (a) uniaxial loading in the 11 direction, (b) uniaxial loading in the 22 direction, (c) shear loading in the 12 direction.

Finally, we assess the nonlinear prediction capability of DMN following the procedure detailed in Chapter 3.7. First, we consider three simple loading-unloading-reloading scenarios (Fig. 3.11). Whether we consider the uniaxial tensile cases (Fig. 3.11a and Fig. 3.11b) or the shear case (Fig. 3.11c), we remark a close agreement between calculations by direct numerical simulations (DNS) and the predictions of DMN. Then, if we consider a more complex scenario in which constraints are applied to all the in-plane strain components (Fig. 3.12a), we notice again the similarity between the results of DNS and those DMN (Fig. 3.12b). Overall, the results reported for this RVE illustrate well the different capabilities of DMN.


doi:10.6342/NTU202301463




**(a)** **(b)**

Figure 3.12: Nonlinear prediction by DMN for a complex loading test. The applied strains are shown in (a) and the corresponding stress responses are shown in (b).



doi:10.6342/NTU202301463



doi:10.6342/NTU202301463

# Chapter 4 A hybrid graph neural network-deep material network model

In this chapter, we will describe in detail our approach to generalizing DMN to arbitrary microstructures under a single graph neural network-deep material network (GNN-DMN) model. First, we pose our problem, which we regard as a broader generalization of DMN. Then, we detail how we regard discretized meshes of microstructures as graphs. Following this, we describe the process which we follow to compress these graphs into feature vectors of meaningful information. A second section of the model then learns to generate appropriate DMNs from these features. The thus created DMNs are stand-alone networks, not necessitating the antecedent parts of the model, and thus can be used for quick predictions during inference.

## 4.1 Problem Formulation

The results reported by several papers [10, 11, 28–30] substantiate the link between the parameters learned by DMN and the geometry of microstructures. For this reason, we postulate that the weights and rotation angles in any trained DMN model can be expressed


doi:10.6342/NTU202301463

as functions of the physical domain $\Omega$ of the microstructure it represents. For simplicity, we utilize a flattened representation of the DMN parameters such that

$$\boldsymbol{p} = \left[ z_N^{k=1,2,\ldots,2^N} \,||\, \theta_{h=0,1,\ldots,N}^{k=1,2,\ldots,2^h} \right] \tag{4.1}$$

in which $\boldsymbol{p}$ is a vector of all the DMN parameters. The symbol $||$ denotes concatenation. Based on our postulate, we may write

$$\boldsymbol{p} = \boldsymbol{p}(\Omega, \boldsymbol{\chi}) \tag{4.2}$$

in which $\boldsymbol{\chi}$ is a term that is included to enable the account of additional variables when determining the parameters. Hence, the homogenization function of DMN (eq. (3.18)) can be formulated as

$$\bar{\boldsymbol{D}} = \mathcal{M}(\boldsymbol{D}^1, \boldsymbol{D}^2 | \, \boldsymbol{p}(\Omega, \boldsymbol{\chi})) \tag{4.3}$$

Equation 4.3 is more general than Eq. 3.18. The form of Eq. 3.18 is rather restrictive: the trainable weights and rotation angles are optimized for a single microstructure during training. In contrast, eq. (4.3) allows them to be functions of the physical domain $\Omega$. This provides an increased degree of flexibility whereby multiple microstructures may be simultaneously trained for. $\mathcal{M}$ takes into account all the key elements affecting the computation of average properties in a homogenization process, namely the physical domain $\Omega$ of the microstructure and the properties of its constituent materials (i.e., $\boldsymbol{D}^1$, $\boldsymbol{D}^2$).

Naturally, the form of eq. (4.3) begs the question as to the class of function suitable for estimating $\boldsymbol{p}$. An obvious requirement for such a class of function is that it should enable the transformation of arbitrary geometries into a compressed, fixed-size representation. In this work, we have elected to employ the utilization of graph neural networks to achieve


doi:10.6342/NTU202301463

this objective.

## 4.2 Model Architecture

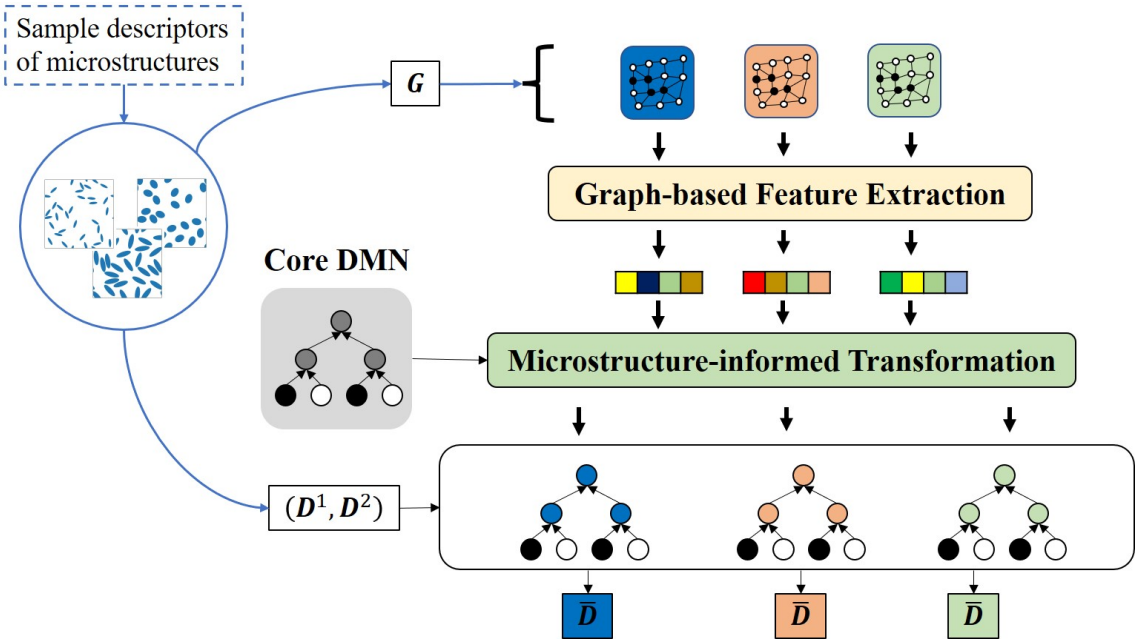



Figure 4.1: Overall architecture of the proposed model.

Our proposed hybrid GNN-DMN model (Fig. 4.1) takes as input the triple $(\boldsymbol{D}^1, \boldsymbol{D}^2, \boldsymbol{G})$, in which $\boldsymbol{G}$ denotes graphs built from discretized meshes of microstructures. The model consists of two segments. The first segment compresses the input graphs $\boldsymbol{G}$ into vectors of useful geometric information. Hence, its objective is the extraction of graph-based features. The second segment utilizes the compressed graph information, as well as parameters from a core DMN model to yield new DMNs suited for each input graph. In other words, it performs a microstructure-informed transformation of the learned features. We will subsequently introduce these two segments.


doi:10.6342/NTU202301463

## 4.3 Graph-based Feature Extraction

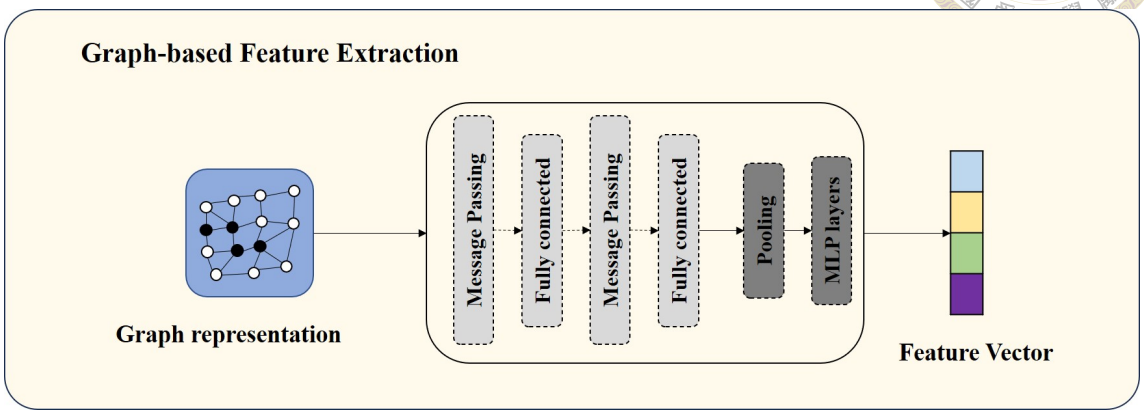



Figure 4.2: First segment of the model through which the graph representations of microstructures are compressed into feature vectors retaining meaningful information about them.

In the first segment (Fig.4.2), the graph data first passes through several layers of message passing and fully connected networks. At this stage, nodes everywhere in the graph gather messages from their neighbors and share information with them as well. Optimally, the features should be projected into higher dimensions to facilitate the acquisition of complex and useful insights.

After several rounds of information sharing, a graph pooling operation is performed. The objective at this stage is to infer *global* graph features from *local* node features. One can resort to an array of choices for graph pooling [26]. The list includes average pooling, max pooling, TopKPooling, DiffPool, etc. In this work, we have simply utilized average pooling.

Following the obtention of graph-level features through pooling, a multi-layer perceptron (MLP) is then used to map them to a feature vector containing meaningful information about the input graph. We may express this step through the following this step


doi:10.6342/NTU202301463

through the following function $\mathcal{H}$

$$\boldsymbol{X}_{feats} = \mathcal{H}(\boldsymbol{G} \mid \Theta_{\mathcal{H}}) \tag{4.4}$$

in which $\boldsymbol{X}_{feats}$ denotes the learned feature vector. $\boldsymbol{G}$ is obviously the input graph to the GNN-based model $\mathcal{H}$. $\Theta_{\mathcal{H}}$ refers to the fitting parameters of $\mathcal{H}$. $\mathcal{H}$ will later be defined by layers of graph neural networks and MLPs. Hence, $\Theta_{\mathcal{H}}$ is, in other words, the set of weights and biases defining these networks.

So far, we have not yet introduced the process for the conversion of meshes into graphs. Thus, we review graph theory in Chapter 4.3.1, and explain how we convert meshes into graphs in Chapter 4.3.2. To conclude this section, we detail the formulation of MLPs and GATv2 [6], which are involved in the treatment of the graph data in Chapter 4.3.3.

### 4.3.1 Graph Theory

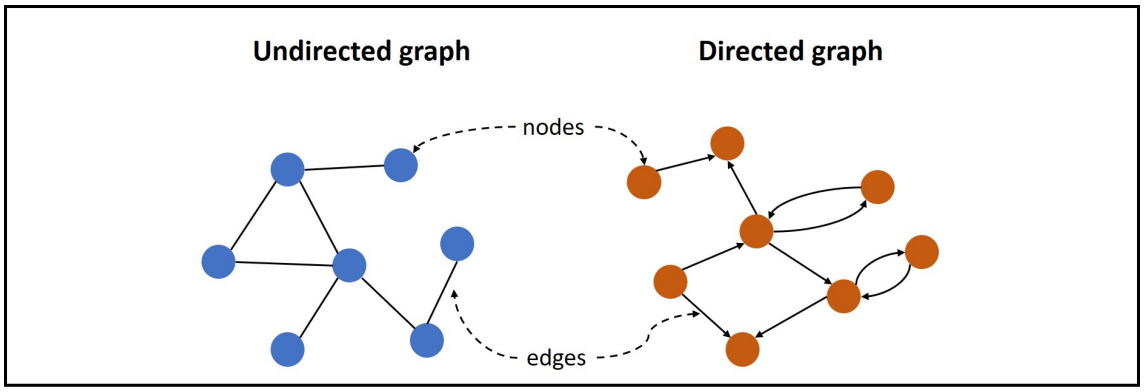



Figure 4.3: A graph is used to model objects (as nodes) and the relationships between them (as edges).

Graph theory is a branch of discrete mathematics that studies the properties of graphs and networks. It covers a wide range of applications, including logistics, social media,


doi:10.6342/NTU202301463

genomics, image analysis, etc. In this context, a graph (Fig. 4.3) is a mathematical representation of a set of objects and their relationships. The set of objects (referred to also as *nodes* or *vertices*) is denoted by $\boldsymbol{V}^G$, while the interactions (*links* or *edges*) between them is represented by the set $\boldsymbol{E}^G$. In mathematical terms, a graph is a two-element tuple $\boldsymbol{G} = (\boldsymbol{V}^G, \boldsymbol{E}^G)$ of vertices

$$\boldsymbol{V}^G = \{v_i^G\}_{i=1}^n \tag{4.5}$$

and edges

$$\boldsymbol{E}^G \subseteq \{\{v_i^G, v_j^G\} \quad | \quad v_i^G, v_j^G \in \boldsymbol{V}^G \wedge v_i^G \neq v_j^G\} \tag{4.6}$$

The inter-connectivity and overall structure of a graph enables to categorize graphs into different types. If one focuses on the directions associated with edges, a graph may be classified as either directed (meaning that each edge has a single direction associated with it) or undirected (edges are bidirectional).

Using graph theory, many problems can naturally be formulated as graphs. For instance, a social network graph can be constructed by regarding individuals as vertices and the existence of friendship between them as edges. A metro system is another example of graph, wherein the nodes are the stations and the links, the presence of a connectivity between them. We will subsequently show how we formulate the physical domains of RVEs as graphs.


doi:10.6342/NTU202301463

### 4.3.2 Meshes as graphs

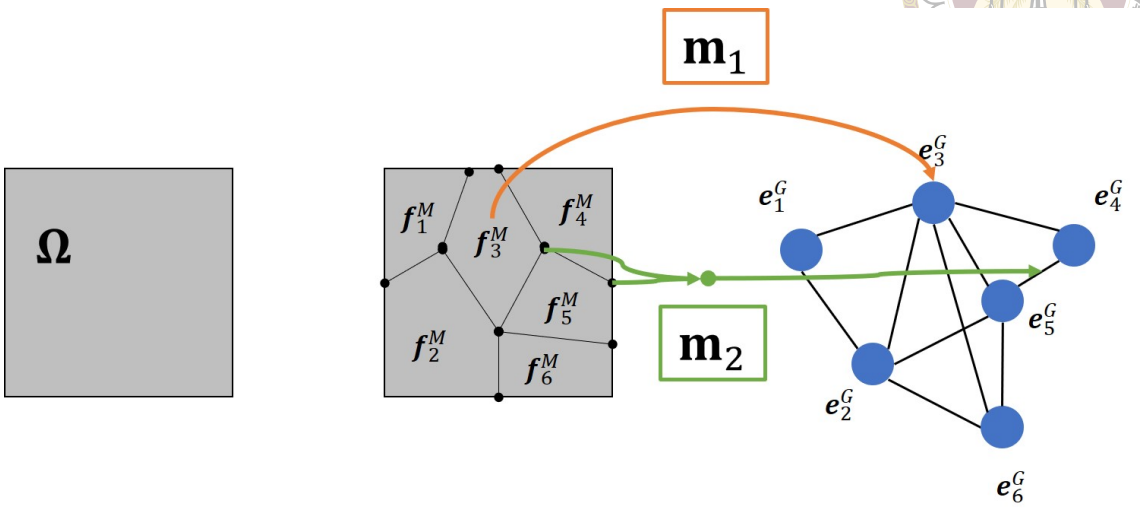



Figure 4.4: Representation of the physical domain $\Omega$ into a graph.

Our objective pertains to the completion of this task: computing the average properties of RVEs. It is a micro-scale boundary-value problem (BVP). Traditionally, the solution to this problem involves approximating the continuous domain of the RVE with a finite set of discrete mesh elements through the process known as *discretization*. The discretization process can be mathematically represented as follows.

$$\Omega \approx \boldsymbol{M} = (\boldsymbol{N}^M, \boldsymbol{E}^M, \boldsymbol{F}^M) \tag{4.7}$$

D1. $\Omega$ is the continuous domain of a PDE,

D2. $\boldsymbol{M}$ is the discretized mesh approximating the domain $\Omega$. It is a triple constituted by a set $\boldsymbol{N}^M$ of nodes, a set $\boldsymbol{E}^M$ of edges, and a set $\boldsymbol{F}^M$ of elements.

Each edge $\boldsymbol{e}_i^M$ in $\boldsymbol{E}^M$ links two nodes in $\boldsymbol{N}^M$. Therefore, $\boldsymbol{e}_i^M$ is a subset of $\boldsymbol{N}^M$.

$$\boldsymbol{e}_i^M = \{n_i^M, n_j^M\} \subset \boldsymbol{N}^M \tag{4.8}$$


doi:10.6342/NTU202301463

Likewise, each mesh element $\boldsymbol{f}_i$ in $\boldsymbol{F}^M$ is uniquely defined by a set of nodes in $\boldsymbol{N}^M$.

$$\boldsymbol{f}_i = (n_i^M, .., n_j^M) \quad | \quad \{n_i^M, ..., n_j^M\} \subset \boldsymbol{N}^M \tag{4.9}$$

The cardinality of $\boldsymbol{f}_i$ is indicative of the type of element it belongs to. In 2D for instance, $|\boldsymbol{f}_i| = 4$ for linear quadrilateral elements.

We regard the meshes of RVEs as graphs (Fig. 4.4). Our objective of solving for the average properties of those microstructures is then attained by training a neural network architecture whose inputs include their graphs. We transform meshes into graphs by using a correspondence which is detailed as follows. Given the discretized mesh $\boldsymbol{M}$ of the domain $\boldsymbol{\Omega}$, each element $\boldsymbol{f}_i$ within it is regarded as a vertex $\boldsymbol{v}_i$ in a graph $\boldsymbol{G}$. In other words, we resort to the mapping $\boldsymbol{m}_1$,

$$\boldsymbol{m}_1 : \boldsymbol{F}^M \rightarrow \boldsymbol{V}^G \tag{4.10}$$

$\boldsymbol{m}_1$ is a bijection as there is a one-to-one correspondence between $\boldsymbol{F}^M$ and $\boldsymbol{V}^G$. Each vertex in the graph is paired with exactly one element in the mesh. Conversely, each element in the mesh corresponds to a graph vertex.

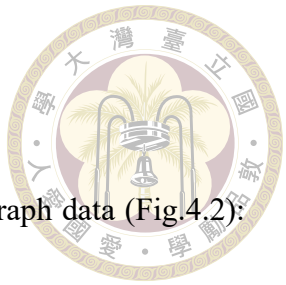
If any two elements respectively $\boldsymbol{f}_i$ and $\boldsymbol{f}_j$ are adjacent in the mesh (i.e., $\boldsymbol{f}_i \cap \boldsymbol{f}_j \neq \varnothing$), an edge is determined to exist between their corresponding vertices in the graph. This correspondence yields a mapping $\boldsymbol{m}_2$

$$\boldsymbol{m}_2 : \boldsymbol{N}^M \rightarrow \boldsymbol{E}^G \quad | \quad \forall \boldsymbol{e}_i^G \in \boldsymbol{E}^G, \exists \boldsymbol{n}_j^M \in \boldsymbol{N}^M, \boldsymbol{m}_2(\boldsymbol{n}_j^M) = \boldsymbol{e}_i^G \tag{4.11}$$

Unlike $\boldsymbol{m}_1$, $\boldsymbol{m}_2$ is a surjection. This is because, for two vertices in the graph sharing an edge, their corresponding element in the mesh may share more than one node.


doi:10.6342/NTU202301463

### 4.3.3 Treatment of graph data



Two types of ML models are involved in the treatment of the graph data (Fig.4.2): multi-layer perceptrons and graph neural networks.

#### 4.3.3.1 Multi Layer Perceptron

In a MLP, an input layer, one (or many) hidden layer(s), and an output layer are sequentially stacked together with so-called activation functions to create a nonlinear mapping between inputs and outputs. Let $\boldsymbol{h}_{i-1} \in \mathbb{R}^{S \times F}$ be the input to layer $i$ in a MLP where $S$ is the number of samples, and $F$ the number of features for each sample. The output $\boldsymbol{h}_i$ of this layer is given by

$$\boldsymbol{h}_i = act\left(\boldsymbol{W}_i^T \boldsymbol{h}_{i-1} + \boldsymbol{b}_i\right) \tag{4.12}$$

with the weight matrix $\boldsymbol{W}_i \in \mathbb{R}^{F_i \times F_{i-1}}$ and bias $\boldsymbol{b}_i \in \mathbb{R}^{F_i}$. $\cdot^T$ represents transposition; $act(\bullet)$ is an activation function such as ReLU or Softplus.

#### 4.3.3.2 Message Passing in Graph Neural Networks

Graph neural networks (GNNs) are tailored for graph-structured data. They learn to represent information about graphs by updating the features of each node based on the features of its neighboring nodes and edges. This process is otherwise known as *message passing* between the nodes. Several GNN models have been proposed over the years, and they mostly differ in the way the message passing is carried out. In virtually all variants of GNNs, the message passing step makes use of a modified form of Eq. 4.12. In this work, we make use of GATv2 [6]. We have selected it for its expressive dynamic attention, and superior performance [6] over other popular counterparts such as the GCN [23],


doi:10.6342/NTU202301463

GraphSAGE [16], GAT [44]. We detail below the message passing process in GATv2.

Let us consider a graph $\boldsymbol{G} = (\boldsymbol{V}^G, \boldsymbol{E}^G)$. Each vertex $v_i^G$ in the set $\boldsymbol{V}^G$ is associated with a feature vector $h_l \in \mathbb{R}^F$. The stacking of all the feature vectors leads to the set of node features $\boldsymbol{h} = \{h_l\}_{l=1}^n \in \mathbb{R}^{n \times F}$. The objective is to obtain, after message passing, a new set of node features $\boldsymbol{h}' \in \mathbb{R}^{n \times F'}$. Within this step, GATv2 makes use of attention coefficients $\alpha_{ij}$ to discriminate between the relevancy of different nodes. Given node $v_i^G$ and its neighborhood $\mathcal{N}_i$. The attention coefficients between $v_i^G$ and all nodes $v_j^G \in \mathcal{N}_i$ are computed using the scoring mechanism $e : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \to \mathbb{R}$

$$e(\boldsymbol{h}_i, \boldsymbol{h}_j) = \boldsymbol{a}^T LeakyReLU\left(\boldsymbol{W} \cdot [\boldsymbol{h}_i || \boldsymbol{h}_j]\right) \tag{4.13}$$

$\boldsymbol{W}$ is a learnable weight matrix as encountered in equation 4.12. $\boldsymbol{a} \in \mathbb{R}^{2F'}$ is a weight vector whose optimal values have to be trained as well. The symbol $||$ represents vector concatenation.

After normalization through the softmax function, the final attention coefficients $\alpha_{ij}$ are obtained as

$$\alpha_{ij} = softmax_j\left(e(\boldsymbol{h}_i, \boldsymbol{h}_j)\right) = \frac{\exp\left(e(\boldsymbol{h}_i, \boldsymbol{h}_j)\right)}{\sum_{v_k^G \in \mathcal{N}_i} \exp\left(e(\boldsymbol{h}_i, \boldsymbol{h}_k)\right)} \tag{4.14}$$

Given $\alpha_{ij}$ all computed, the node features in a given neighborhood are weighted based on their relative importance and aggregated to obtain the updated features at each node using

$$h_i' = act\left(\sum_{v_j^G \in \mathcal{N}_i} \alpha_{ij} \boldsymbol{W} h_j\right) \tag{4.15}$$

Equation (4.15) constitutes the message passing mechanism in GATv2. GATv2 further


doi:10.6342/NTU202301463

allows the flexibility of using multi-head attention. In such a case, the aggregated features are instead obtained using

$$h_i' = act\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{v_j^G \in \mathcal{N}_i} \alpha_{ij}^k \boldsymbol{W}^k h_j\right) \tag{4.16}$$

Here $K$ denotes the number of attention heads. This concludes the bulk of message passing in GATv2.

#### 4.3.3.3 Graph Features

Based on the formulation of GATv2, each node in the graph $\boldsymbol{G}$, representing an element in the mesh $\boldsymbol{M}$, is associated with a vector of node features $h_l$. In this work, we have considered four such features, which we deem relevant to the task of modeling composite microstructures:

1. the area of the element in the mesh
2. the $xy$ coordinate of the centroid of that element
3. the phase identifier of the element, i.e., matrix or inclusion
4. whether the element is touching the border of the RVE. This is to take into account the periodicity of RVEs.

This feature selection exercise can be easily tailored for other families of microstructures. In the case of polycrystals for instance, one may select grain size and crystal orientation as graph node features.


doi:10.6342/NTU202301463

## 4.4 Microstructure-informed Transformation

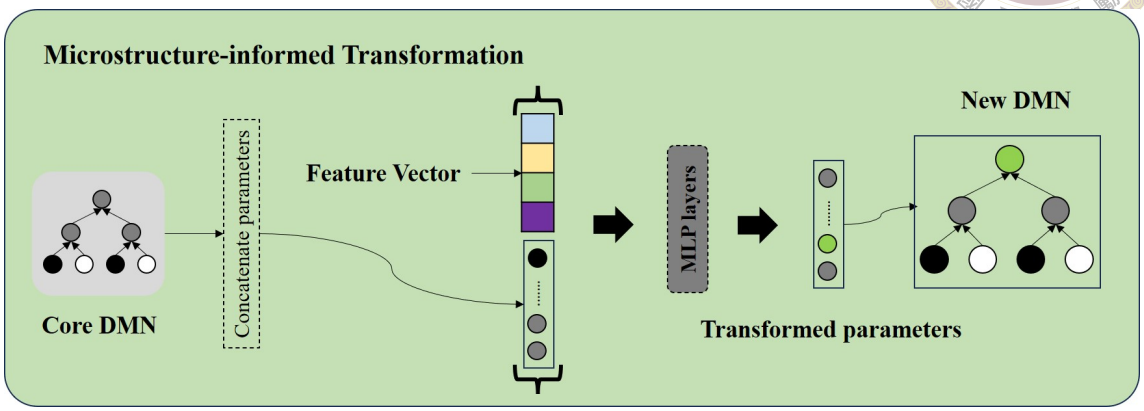



Figure 4.5: Second segment of the model in which the information from learned feature vectors are leveraged to generate new DMNs.

In the second segment (Fig. 4.5), we utilize the feature vector $\boldsymbol{X}_{feats}$ (eq. (4.4)) extracted from the input graph. This feature vector is used in parallel with a *core DMN* whose parameters will be optimized during training. One can think of the core DMN as a reference network that facilitates the creation of new DMNs. In light of eq. (4.2), it can be seen as the extra variable $\boldsymbol{\chi}$ influencing the determination of new DMN parameters. The depth of its tree is chosen in relation to the complexity of the problem at hand. Previous works [29, 30] have shown that a DMN of 5 layers should be appropriate for the category of microstructures which we will be considering. Thus, we set the depth of the core DMN to 5. The total number of parameters in this core DMN are $3 \times 2^5 - 1 = 95$, while the dimension of the feature vector $\boldsymbol{X}_{feats}$ is 32.

A dense MLP network then reads the compressed feature vector $\boldsymbol{X}_{feats}$ and a flattened representation of parameters from the core DMN model. After several layers of nonlinear mappings, it transforms these inputs into a vector representing parameters for a newly derived DMN tuned for the input graph data. This transformation may be expressed


doi:10.6342/NTU202301463

as

$$\tilde{\boldsymbol{p}} = \mathcal{T}(\boldsymbol{X}_{feats}, \bar{\boldsymbol{p}} \mid \Theta_{\mathcal{T}}) \tag{4.17}$$

where $\bar{\boldsymbol{p}}$ refer to parameters of the core DMN model. $\Theta_{\mathcal{T}}$ represents the fitting parameters of the MLP network denoted by $\mathcal{T}$; these parameters are weights and biases defining the MLP. $\tilde{\boldsymbol{p}}$ constitutes the parameters of the derived DMN, and has the same dimension as $\bar{\boldsymbol{p}}$. The thus obtained DMN is a stand-alone model. It can take any $(\boldsymbol{D}^1, \boldsymbol{D}^2)$ pair and predict the homogenized tangent $\bar{\boldsymbol{D}}$ for the corresponding input graph data $\boldsymbol{G}$. By reference to Eq. 4.3, we may write the prediction of this DMN as

$$\bar{\boldsymbol{D}} = \mathcal{M}(\boldsymbol{D}^1, \boldsymbol{D}^2 \mid \tilde{\boldsymbol{p}}) \tag{4.18}$$

However, unlike the original formulation of DMN in which the parameters are directly fitted (Eq. 3.18), they are here predicted

$$\tilde{\boldsymbol{p}} = \mathcal{T}(\mathcal{H}(\boldsymbol{G} \mid \Theta_{\mathcal{H}}), \bar{\boldsymbol{p}} \mid \Theta_{\mathcal{T}}) \tag{4.19}$$

Simplifying eq. (4.19), we may write

$$\tilde{\boldsymbol{p}} = \tilde{\boldsymbol{p}}(\boldsymbol{G} \mid \Theta_{\mathcal{H}}, \Theta_{\mathcal{T}}, \bar{\boldsymbol{p}}) \tag{4.20}$$

By using eq. (4.20), we may expand eq. (4.18) into

$$\bar{\boldsymbol{D}} = \mathcal{M}(\boldsymbol{D}^1, \boldsymbol{D}^2 \mid \tilde{\boldsymbol{p}}(\boldsymbol{G} \mid \Theta_{\mathcal{H}}, \Theta_{\mathcal{T}}, \bar{\boldsymbol{p}})) \tag{4.21}$$

Solely considering the input data and free parameters in eq. (4.21), we may write the


doi:10.6342/NTU202301463

function $\mathcal{M}$ for the entire model as

$$\bar{\boldsymbol{D}} = \mathcal{M}(\boldsymbol{G}, \boldsymbol{D}^1, \boldsymbol{D}^2 \mid \Theta_{\mathcal{H}}, \Theta_{\mathcal{T}}, \bar{\boldsymbol{p}}) \tag{4.22}$$

The input to the model are the graph representation $\boldsymbol{G}$ of microstructures and the tangents $\boldsymbol{D}^1, \boldsymbol{D}^2$ of their constituent materials. Its fitting parameters are $\Theta_{\mathcal{H}}$, $\Theta_{\mathcal{T}}$, and $\bar{\boldsymbol{p}}$, which have already been defined.

The final outputs of the model $\mathcal{M}$ are homogenized tangents $\bar{\boldsymbol{D}}$. Besides, the model has intermediary outputs $\tilde{\boldsymbol{p}}$. During training, the entire pipeline of the model is involved. However, during inference, the *feature extraction* and *microstructure-informed transformation* parts of the model are involved only once to obtain the derived parameters $\tilde{\boldsymbol{p}}$. Afterwards, the DMNs defined by $\tilde{\boldsymbol{p}}$ are used as stand-alone models, hence allowing to take advantage of the low-dimensionality of DMN. Naturally, such DMNs can be used, not only for the prediction of linear elastic properties, but also for nonlinear analyses.

## 4.5 Model Training

The optimal parameters $\Theta_{\mathcal{H}}$, $\Theta_{\mathcal{T}}$, and $\bar{\boldsymbol{p}}$ of the model $\mathcal{M}$ are determined through training of the network. To penalize the departures of its predictions from target values, the following cost function, initially introduced by [29] is defined:

$$J_0(\mathcal{M}) = \frac{1}{N_s} \sum_i N_s \frac{\|\bar{\boldsymbol{D}}_i^{pred} - \bar{\boldsymbol{D}}_i^{DNS}\|_F^2}{\|\bar{\boldsymbol{D}}_i^{DNS}\|_F^2} \tag{4.23}$$

$\bar{\boldsymbol{D}}_i^{pred}$ and $\bar{\boldsymbol{D}}_i^{DNS}$ respectively denote values predicted by the model and obtained via DNS. $\|\cdot\|_F$, denoting the Frobenius norm, has already been defined in eq. (3.26). $N_s$ is


doi:10.6342/NTU202301463

the number of samples being considered.

We additionally subject the weights at the bottom layer of the core DMN network to the arbitrary constraint:

$$\sum_{i}^{2^N} \bar{w}_N^i = \alpha \tag{4.24}$$

$\alpha$ is a scalar that controls the magnitudes of the fitted weights. We choose to set it equal to $2^{N-2}$ as in [28–30]. The final cost function applied to the outputs of the model $\mathcal{M}$ is:

$$J(\mathcal{M}) = J_0(\mathcal{M}) + \lambda L(\bar{w}) \tag{4.25}$$

with the scalar $\lambda$ being a hyperparameter to be selected, and $L(\bar{w}_N^k)$ being equal to

$$L(\bar{w}_N^k) = \left( \sum_{k}^{2^N} \bar{w}_N^k - \alpha \right)^2 \tag{4.26}$$

such that

$$J(\mathcal{M}) = \frac{1}{N_s} \sum_{i}^{N_s} \frac{\|\boldsymbol{D}_i^{pred} - \boldsymbol{D}_i^{DNS}\|_F^2}{\|\boldsymbol{D}_i^{DNS}\|_F^2} + \lambda \left( \sum_{k}^{2^N} \bar{w}_N^k - \alpha \right)^2 \tag{4.27}$$

In order to quantify the accuracy of predicted tangents during the training process, the same metric defined in eq. (3.29) is used. Besides, we additionally define a volume fraction error

$$\text{volume fraction error} = \frac{1}{N_m} \sum_{i}^{N_m} \frac{|V_{f_i}^{model} - V_{f_i}^{truth}|}{|V_{f_i}^{truth}|} \tag{4.28}$$

in which $N_m$ represents the number of microstructures, $V_{f_i}^{model}$ represents the volume fraction obtained from created DMNs, and $V_{f_i}^{truth}$ represents the volume fraction which can be computed from information in the RVE mesh.



doi:10.6342/NTU202301463

## 4.6 Generation of Microstructures

We generate RVEs based on the *random sequential adsorption* (RSA) method. Before starting the process, we first select the microstructural descriptors of the RVEs. For instance, we might wish to generate microstructures with circular, fixed-size inclusions embedded in their matrix. In this case, the determinant microstructural descriptor would be the inclusion volume fractions. Then, based on the target descriptors, we determine the required quantity of inclusions to be dispersed in the matrix.

At the initialization of the process, we start with an empty surface of the matrix. Then, we select a particle and randomly place it within that surface. Since this is the first placement of particle, there is no issue with overlapping. We then turn to the next particle and try to place it in a new location. Because of a pre-existing inclusion in the matrix, there exists a probability of overlap with it. If such is the case, a location is resampled until it satisfies our criterion of non-collision. This procedure is repeated until all inclusions have been placed onto the RVE matrix. Besides, we make use of special treatments for particles positioned on the RVE boundary to ensure its periodicity. The corresponding algorithm (Alg. 8) is provided in Appendix B.2.


doi:10.6342/NTU202301463

# Chapter 5 Examples

## 5.1 Example 1: Microstructures with circular inclusions

In this section, we investigate the performance of our model. We consider, in this first example, microstructures with circular inclusions in their matrix.

### 5.1.1 Data Generation

Each microstructure can be globally defined by the total volume fraction $v_f$ of its inclusions. Nevertheless, this descriptor, namely the inclusion volume fraction $v_f$, is not fully representative. Generally speaking, there always exists a randomness in the realization of a RVE based on global descriptors. This randomness is inherently due to the shape (i.e., size, inclination) of the inclusions and their locations. We are considering circles, which have a rotational symmetry of order $\infty$. Therefore, the placement of a circular inclusion is uniquely determined for a given location within the matrix (this would not be true if the inclusion was elliptical, for instance). Considering the aforementioned point, randomness can possibly be induced by the size of the inclusions and their positions; but, we choose to fix their dimensions. Eventually, we remain with a single source of randomness: the distributions of the circular inclusions in the RVE.


doi:10.6342/NTU202301463

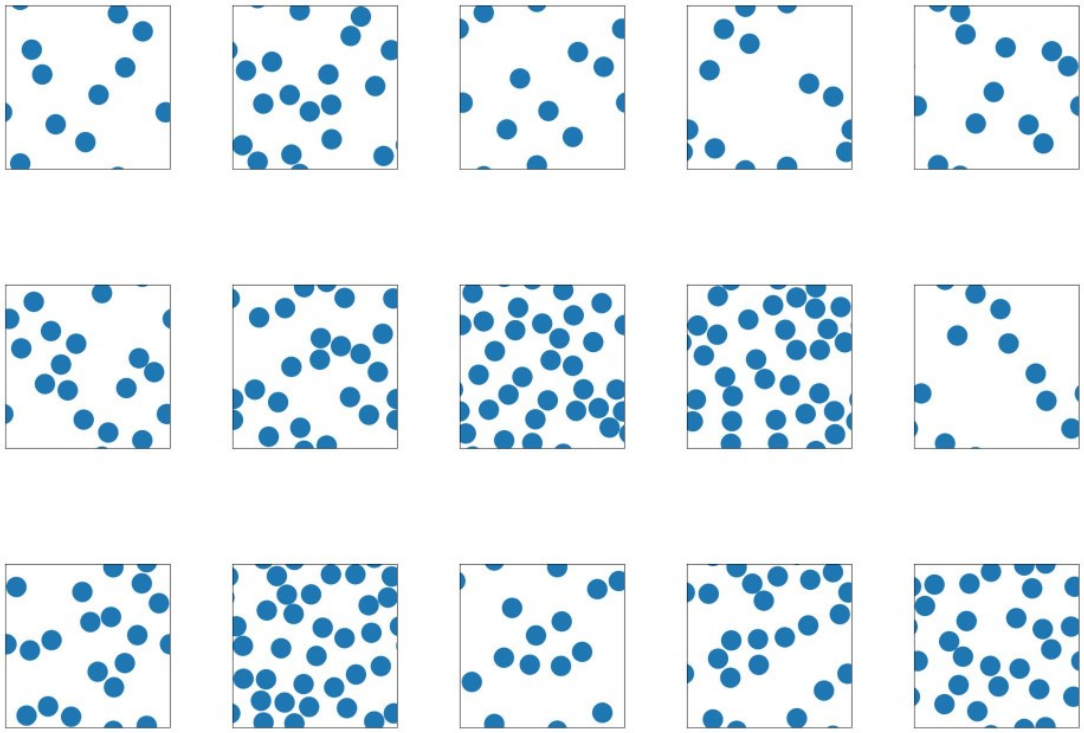


Figure 5.1: Examples of microstructures in the training set.

To generate the microstructures needed for training, we uniformly sample $v_f$, such that $v_f \in U[0.10, 0.40]$. We build two sets of microstructures: the training and validation sets respectively comprised of 200 and 50 RVEs. These considerable number of points sampled within the prescribed volume fraction range give rise to multiple RVEs ultimately sharing the same global descriptors. This can readily be observed in figure 5.1. These microstructures are then discretized into meshes. A mixed triangular-quadrilateral discretization scheme elements has been used in all cases. As to the resolution of the meshes, it is rather fine, the number of elements varying from $\sim 7,000$ to $\sim 10,000$.

Beside the meshes, elastic tangents are also required as inputs within our proposed framework. Consequently, we generate 300 triples $(\boldsymbol{D}^1, \boldsymbol{D}^2, \bar{\boldsymbol{D}})$ for each microstructure via direct numerical simulations (DNS). The material properties used for obtaining $\boldsymbol{D}^1$ and


doi:10.6342/NTU202301463

$\boldsymbol{D}^2$ are the same as those reported in Fig. 3.8. We recall the corresponding details. The constituent phases of the RVEs are assumed to be linear isotropic. The elastic moduli and Poisson's ratios dictating the values in $(\boldsymbol{D}^1, \boldsymbol{D}^2)$ were sampled as follows. The Young's modulus $E^1_{11}$ of phase 1 was held equal to unity, i.e. $log_e(E^1_{11}) = 0$. In contrast, $E^2_{11}$ was allowed to vary such that $-5 \leq log_e(E^2_{11}) \leq 5$. In other words, its magnitude could be $\sim 150$ times as high (or as low) as $E^1_{11}$. The Poisson's ratio for both phases was allowed to vary between 0 and 0.5. As to the target tangents $\bar{\boldsymbol{D}}$, the procedure followed to compute them is as described in Chapter 3.4.

### 5.1.2 Offline Training

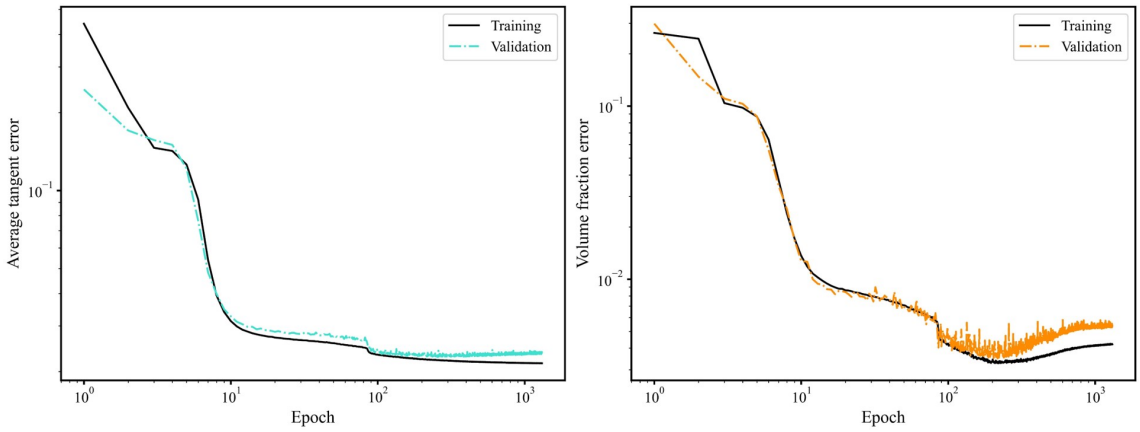



Figure 5.2: Evolution of the error on the predicted tangents and volume fractions during training.

Next, we proceed to train our GNN-DMN model whose core DMN has 5 layers. We do so for over 1000 epochs (Fig 5.2). The computed errors for the training set decreased to 2.15% at the end of the fitting process. The model showed a comparable performance on the validation set, for which the error was evaluated at 2.34%. Not only did the model demonstrate its ability to accurately predict the homogenized tangents, it also display its capacity at capturing the topological information of the meshes taken as inputs. We notice


doi:10.6342/NTU202301463

errors of respectively $0.42\%$ and $0.53\%$ on the predicted volume fractions by the DMNs the model created for microstructures in the training and validation sets.

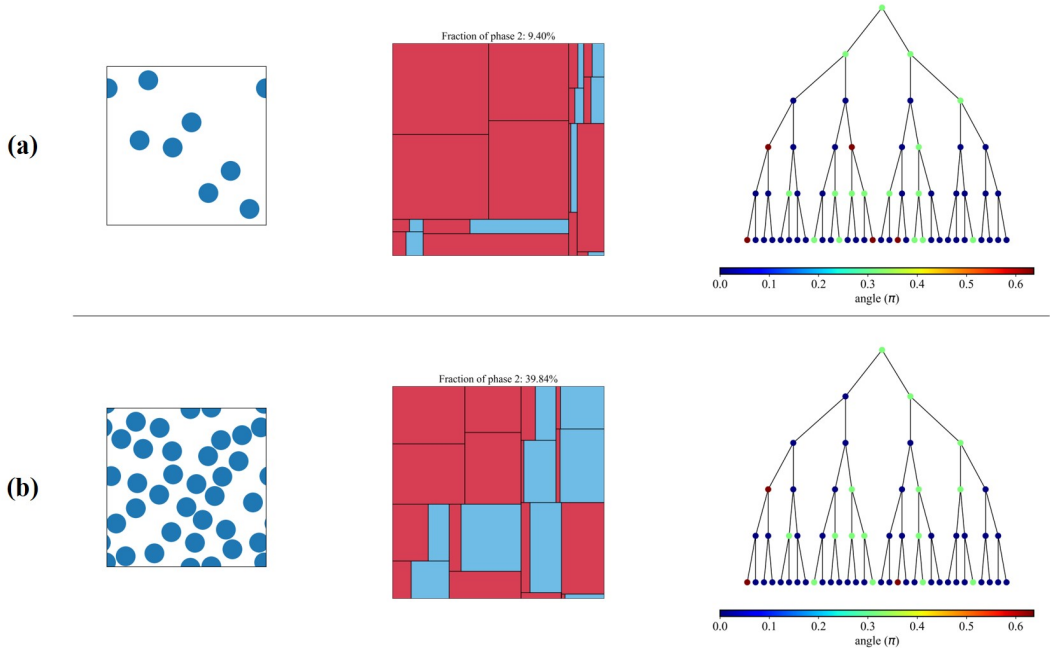



Figure 5.3: Visualization of the DMN parameters generated by the network for two different microstructures.

For illustration purposes, we show a visualization of the treemaps and rotation angles of DMNs generated for two microstructures in the validation set, with inclusion volume fractions at the extremities of the sampling range. These RVEs respectively have inclusion volume fractions of 9.76% (Fig. 5.3a) and 39.02% (Fig. 5.3b). The corresponding DMNs exhibit volume fractions of 9.40% and 39.84%, which are close to the targets. As to the rotation angles, we notice different distributions of them in the tree.


doi:10.6342/NTU202301463

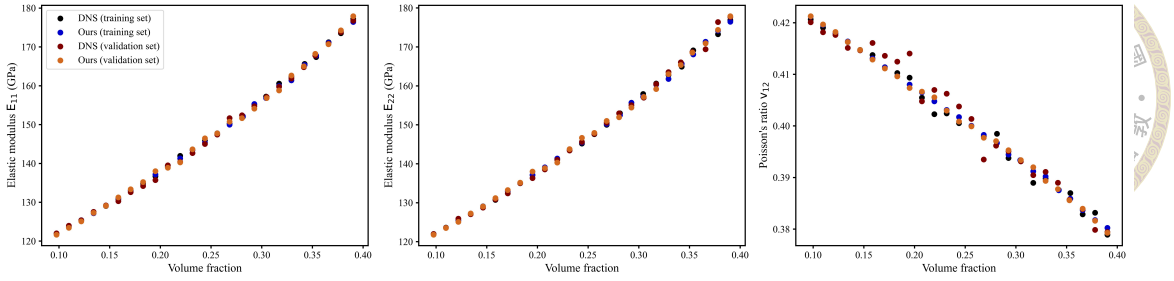



Figure 5.4: Predicted (a) longitudinal Young' s modulus $E_{11}$, (b) transverse Young' s modulus $E_{11}$ , and (c) Poisson' s ratio $v_{12}$ for microstructures in the training and validation set.

### 5.1.3 Online Prediction: Elastic Properties

| | **Young's modulus** $E$ | **Poisson's ratio** $v$ | **Yield stress** $\sigma_y$ | **Tangent modulus** |
|---|---|---|---|---|
| **Phase 1** | 100 GPa | 0.3 | 0.1 GPa | 5 GPa |
| **Phase 2** | 500 GPa | 0.19 | - | - |

Table 5.1: Selected material properties.

The end-goal of the model is to be used in practice as a surrogate to traditional simulation tools in the evaluation of material properties. In light of this consideration, we select a set of material properties, listed in Table 5.1. We do the following. For all the microstructures (200 in the training set and 50 in the validation set), we carry out DNS based on those properties. At first, we solely consider homogenized elastic properties, namely the longitudinal Young's modulus $E_{11}$, transverse Young's modulus $E_{22}$ and Poisson's ratio $v_{12}$. These results are plotted in Fig 5.4a. In parallel, we use the trained model to predict the same properties, also shown in Fig 5.4. We recall that, for each volume fraction, there exists several RVE realizations. Thus, the results reported in Fig 5.4 are the average predictions for each volume fraction. Upon close inspection of the results, we notice the similarity between the predictions and the reference results.

But, we wish to examine the the variability in the predictions yielded by different


doi:10.6342/NTU202301463

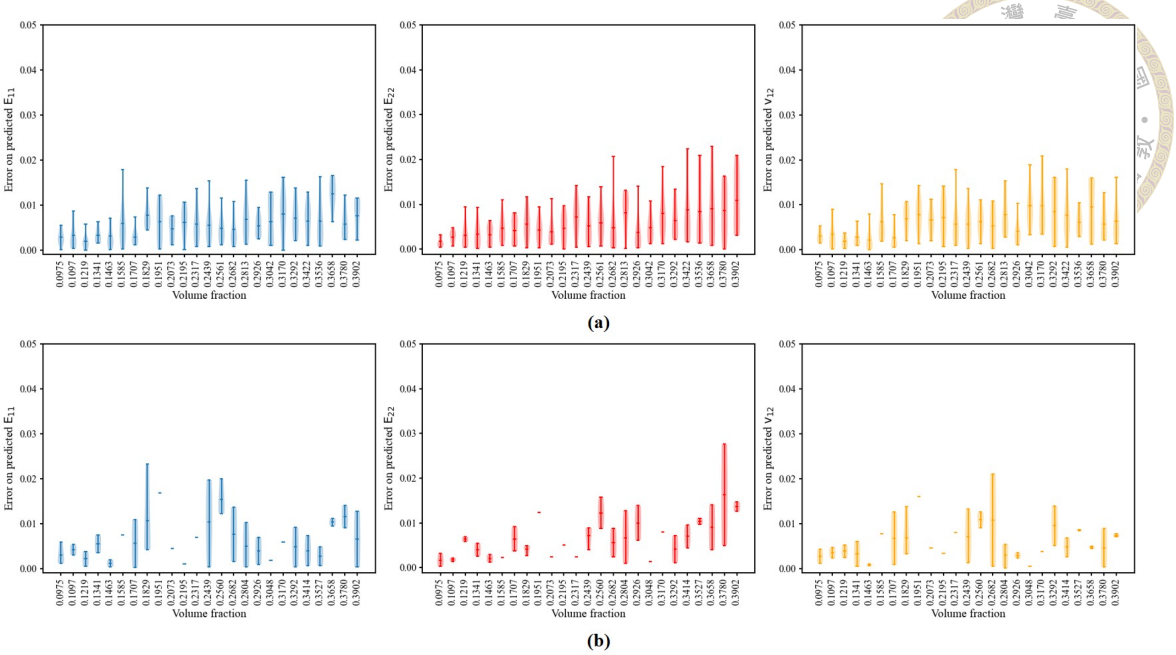



Figure 5.5: Relative errors on the predicted longitudinal Young's modulus $E_{11}$, transverse Young's modulus $E_{11}$, and Poisson's ratio $\upsilon_{12}$ for microstructures in the (a) training set and (b) validation set.

RVE realizations for a given volume fraction. Hence, we show related statistics in Fig 5.5. As in Fig 5.4, the x-axis of the plots indicates the inclusion volume fractions. However, unlike Fig 5.4 in which the predictions for each RVE realization was directly plotted, Fig 5.5 shows a violin plot of the errors for each of those volume fractions. The bottom, middle and top bar in each violin plot respectively denote the minimum, mean and maximum errors for the different RVE realizations of the considered volume fraction. Results pertaining to microstructures in the training set are shown in Fig 5.5a, while those for microstructures in the validation set are given in Fig 5.5b. One observation is conspicuous: whether the training or validation microstructures are considered, the mean errors on the predicted properties are predominantly inferior to 1%.


doi:10.6342/NTU202301463

### 5.1.4 Online Prediction: Nonlinear Responses

Next we turn to the performance of the model when it comes to the nonlinear response of the RVEs. We consider the same microstructures showed in Fig. 5.3. For each of them, we perform three (3) loading-unloading-reloading tests using all the properties listed in Table 5.1. Two of them are uniaxial tests in the two orthogonal directions (i.e., 1 and 2), while the last one is an in-plane shear test. The results for the lower and higher volume fraction microstructures are respectively reported in Fig 5.6a and Fig 5.6b. From close inspection, we notice the close agreement between the responses predicted by the model and those determined via DNS. To be able to quantify this performance, we define in a manner similar to [21] errors on the predicted stresses, as:

$$\text{mean-relative error} = \frac{\frac{1}{n}\sum_{i=1}^{n} |\sigma_i^{DNS} - \sigma_i^{model}|}{\max_{i=1,\dots,n} |\sigma_i^{DNS}|} \tag{5.1}$$

$$\text{max-relative error} = \frac{\max_{i=1,\dots,n} |\sigma_i^{DNS} - \sigma_i^{model}|}{\max_{i=1,\dots,n} |\sigma_i^{DNS}|} \tag{5.2}$$

Based on the definitions given in equations 5.1 and 5.2, the mean-relative errors for the three tests in Fig 5.6a are $1.32\%, 0.82\%, 0.34\%$, while their max-relative errors are $5.40\%, 3.00\%, 1.69\%$. By comparison, the mean-relative errors for Fig 5.6b are $2.11\%, 2.52\%, 1.01\%$, while the max-relative errors are $7.77\%, 6.89\%, 3.51\%$.

Given that we have a visual representation of what the computed errors on the stress responses look like, we repeat the same tests for all the microstructures in the validation set. Fig 5.7 displays histograms of the computed errors for the nonlinear response of those 50 microstructures. Considering the mean-relative error, the mean of the distributions for



doi:10.6342/NTU202301463

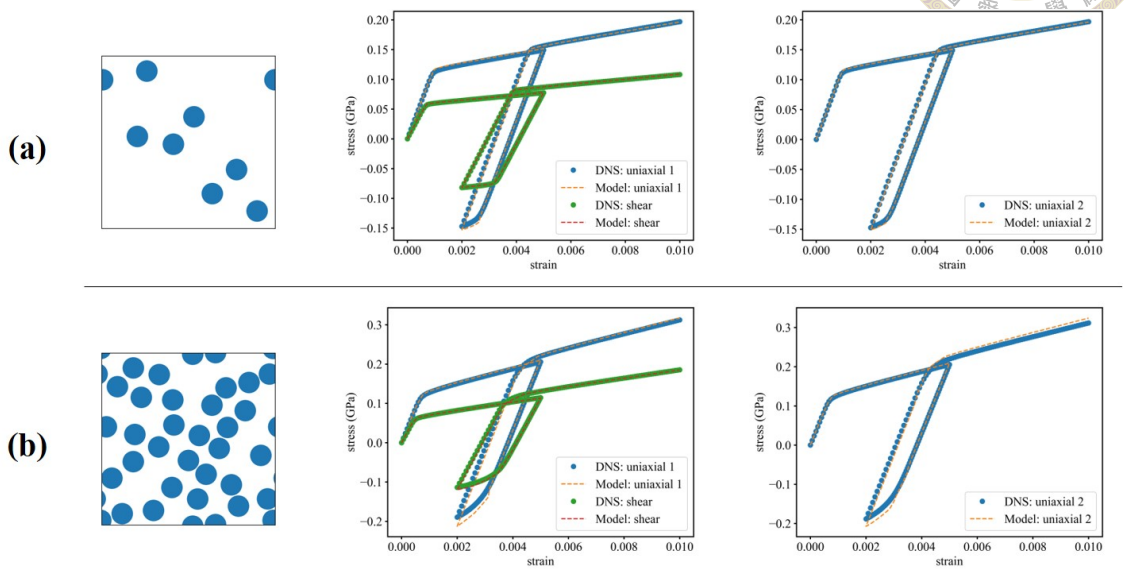



Figure 5.6: Nonlinear prediction with interpolated DMN models for selected microstructures in the validation set. The properties used are those listed in Table 5.1.

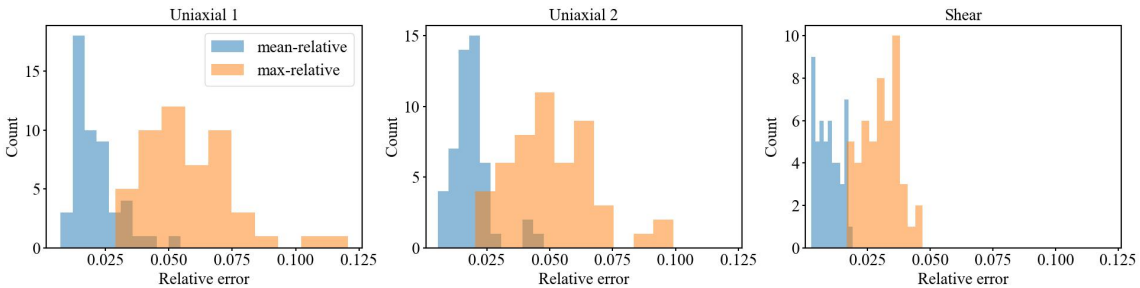



Figure 5.7: Distribution of the errors yielded by the created DMN models for the nonlinear prediction performed based on the properties listed in Table 5.1.



doi:10.6342/NTU202301463

the three tests are respectively 2.06%, 1.91% and 0.95%. As to the max-relative error, the mean of the distributions are 5.78%, 5.06% and 3.03%. Comparing with the statistics of Fig 5.6, these are reliable results. Our model shows its capability to generalize well on this family of RVE.

## 5.2 Example 2: Microstructures with elliptic inclusions

The previous example proved the capacity of our model to learn the micromechanics of the family of RVEs with circular inclusions. Here, we attack the more complex case of microstructures with multi-dimensional descriptors.

### 5.2.1 Data Generation

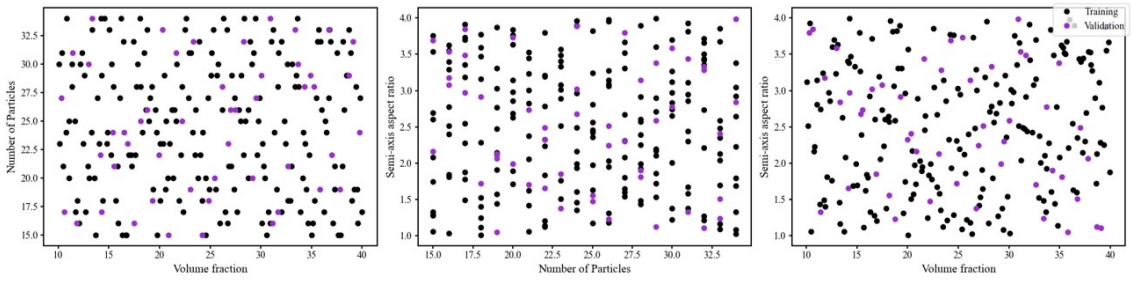



Figure 5.8: Sampling space of descriptors (i.e., volume fraction, number of particles, semi-axis ratio) for the RVEs. The training and validation set respectively comprise 200 and 50 RVEs.



doi:10.6342/NTU202301463

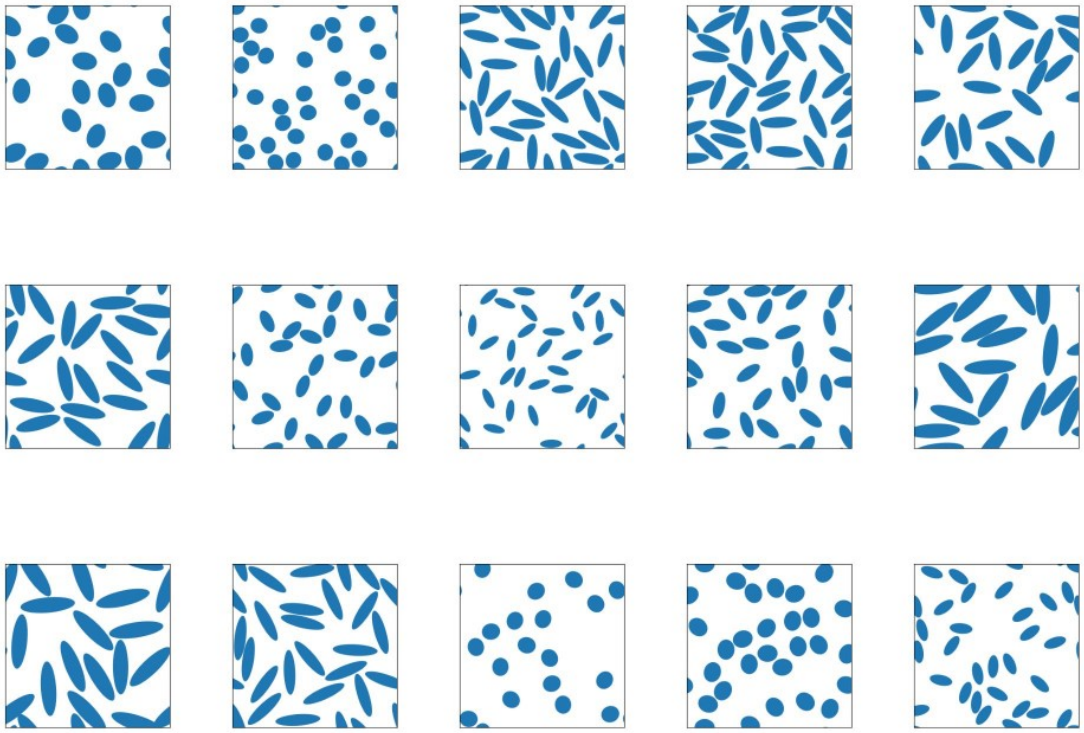


Figure 5.9: Examples of RVEs in the training set. We notice the variety in the shape and size of inclusions in the microstructures.

We consider RVEs with inclusions of varying quantities, shapes, sizes. Thus, we define a three-dimensional descriptor space specified by the number of particles $N_p$ in each RVE, the semi-axis aspect ratio $A_r$ of these ellipse-shaped particles, as well as the volume fraction $v_f$ occupied by the particles. The actual size of the particles in the RVEs are deduced from $N_p$ and $v_f$. As to the orientations of the particles during their placement in the RVEs, they are assigned at random. Based on these descriptors, we respectively 200 microstructures for the training set and 50 for the validation set. The corresponding space of descriptors obtained through latin hypercube sampling is shown in Fig 5.8. Fig 5.9 shows examples of RVEs thus obtained in the training set. We readily notice the diversity among them. The microstructures exhibit a variation in the sizes, shapes and distribution of their inclusions. As in the first example, we generate 300 $(\boldsymbol{D}^1, \boldsymbol{D}^2, \bar{\boldsymbol{D}})$ for each RVE.


doi:10.6342/NTU202301463

The material properties used in this case are the same as shown in Fig 3.8.

### 5.2.2 Offline Training

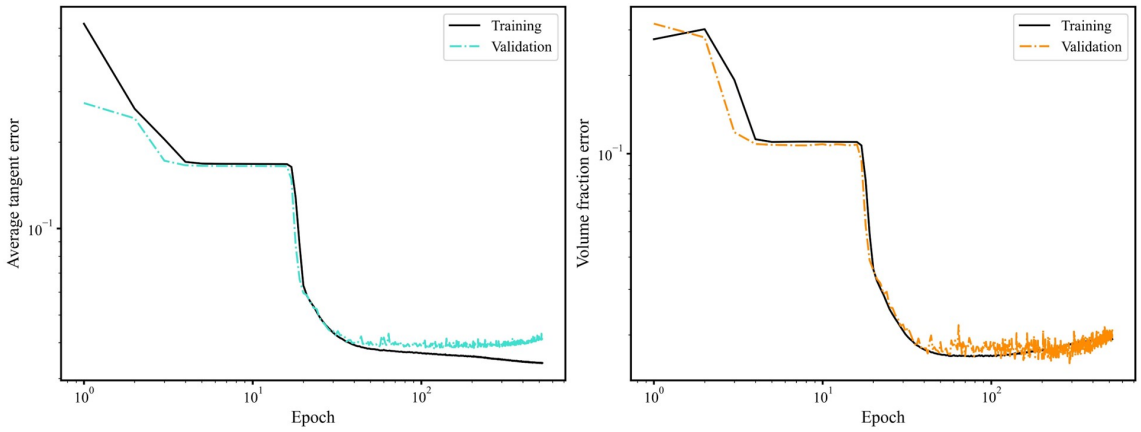



Figure 5.10: Evolution of the errors on the predicted tangents and volume fractions during training.

We then proceed to train the GNN-DMN model, with a core DMN of 5 layers, for over 500 epochs. The model started to display some overfitting behavior on the predicted tangents (Fig 5.10). As a result, we interrupted the training earlier than in the previous case. At this point, the model yielded an average error of 3.40% for RVEs in the training set, and 4.18% for those in the validation set. The model is even more accurate on the volume fractions inferred from its generated DMNs: 1.93% and 2.13% for the training and validation set.

For illustration purposes, we select two very dissimilar RVEs from the training set. The first one (Fig 5.11a) has inclusions of considerable size, with high aspect ratio, and a volume fraction of 37.74%. In contrast, the second one (Fig 5.11b) is a RVE whose inclusions are minuscule, almost circular, and has a volume fraction of 13.08%. Considering the parameters of their corresponding DMNs, we notice that the rotation angles are unlike. The volume fractions of the DMNs illustrated by the treemaps are respectively


doi:10.6342/NTU202301463

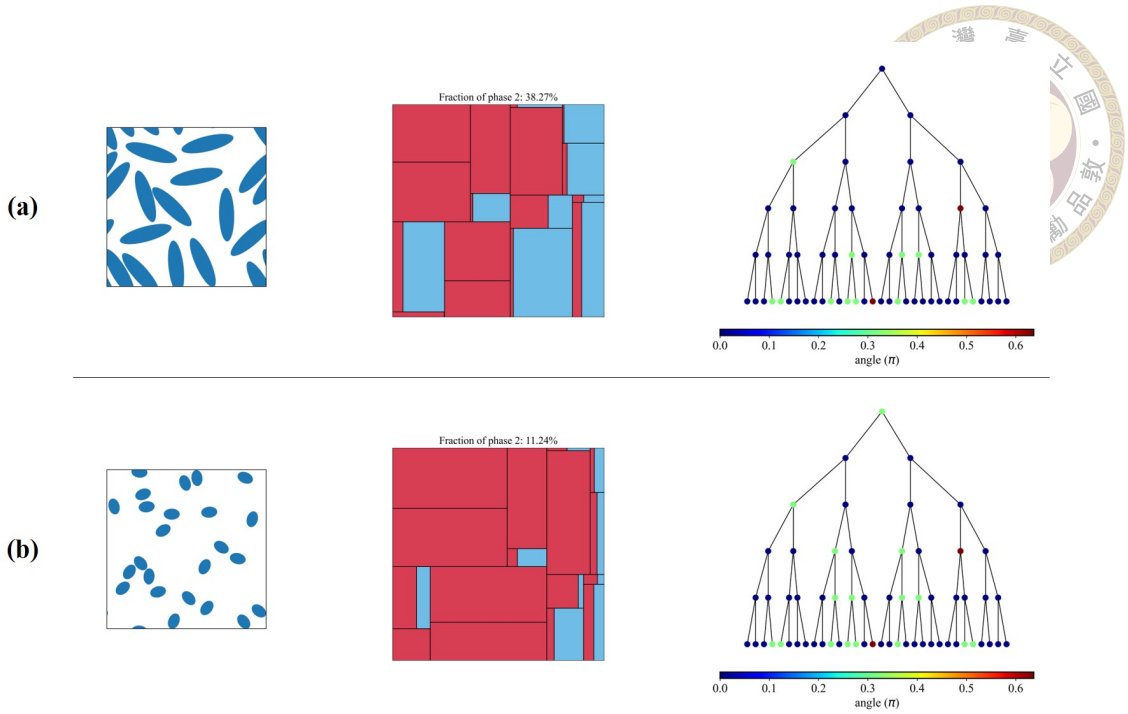



Figure 5.11: Visualization of the DMN parameters generated by the network for two different microstructures.

38.27% and 11.24%. Although the predicted volume fraction is less accurate in the second case, it is nonetheless surprising to notice the ability of the model to generate adequate DMNs, when taking into account the diversity of microstructures in the data sets.

### 5.2.3 Online Prediction: Elastic Properties

We then proceed to scrutinize the performance of this newly fitted model in predicting the homogenized properties of RVEs in the training set using the elastic constants listed in Table 5.1. We utilize 3D plots (Fig 5.12) whose axes correspond to the space of RVE descriptors. The color of each point indicates the result for the RVE it represents. Fig 5.12a shows the predicted longitudinal $E_{11}$, transverse $E_{22}$ Young's moduli, and Poisson's ratio $v_{12}$ obtained via DNS. Fig 5.12b shows the same results, as predicted by our model. A visual inspection indicates a good similarity between the two groups. To be more rigorous, we quantify the corresponding errors, which are reported Fig 5.13a. The mean of


doi:10.6342/NTU202301463

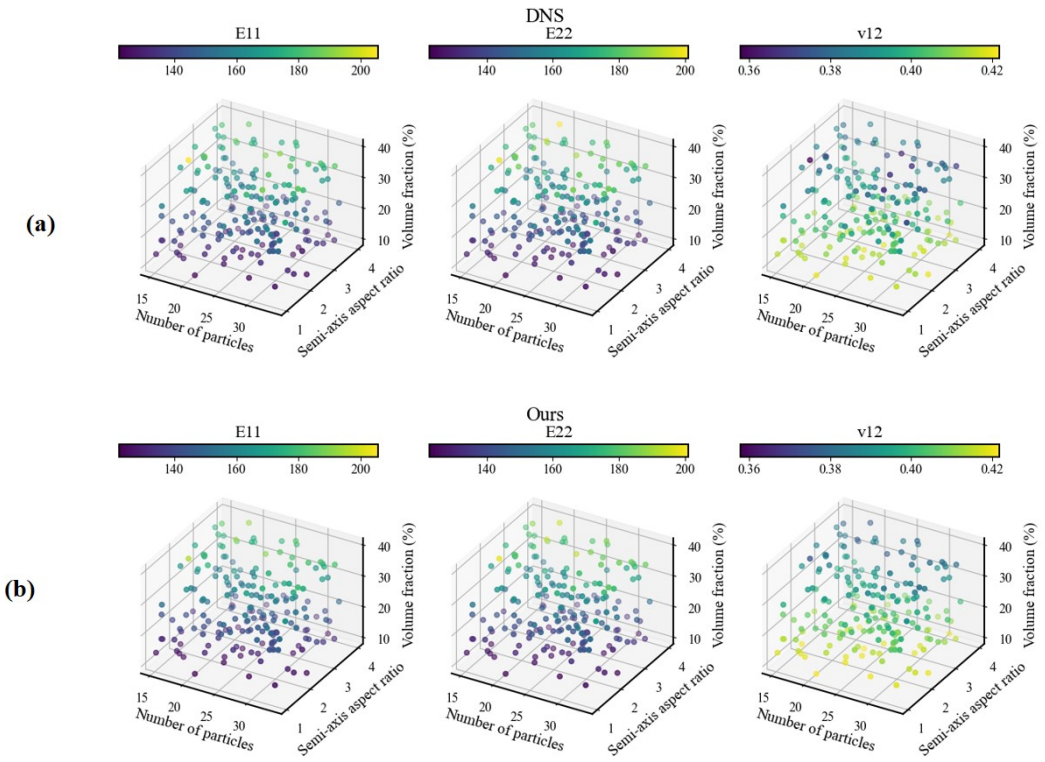



Figure 5.12: Predicted longitudinal Young's modulus $E_{11}$, transverse Young's modulus $E_{22}$ and Poisson's ratio $v_{12}$ for microstructures in the training set (a) via direct numerical simulation and (b) by our model.

the relative errors for $E_{11}$, $E_{22}$, and $v_{12}$ are respectively $1.51\%$, $1.32\%$ and $1.01\%$. We additionally consider microstructures in the validation set (Fig 5.13b). The averages for the same set of relative errors are $1.42\%$, $1.44\%$, and $0.97\%$.

### 5.2.4 Online Prediction: Nonlinear Responses

Last but not least, we must evaluate the use of the model for nonlinear predictions. We utilize the microstructure shown reported in Fig 5.11. Their corresponding DMNs predict stress-strain curves anologous to the results of DNS. For reference, the mean-relative errors (Eq. 5.1) for the predictions in Fig 5.14a are $2.57\%$, $0.64\%$, and $0.66\%$ respectively for the uniaxial 1, uniaxial 2 and shear tests. Meanwhile, the max-relative errors (Eq. 5.2) are $7.24\%$, $3.88\%$, and $3.24\%$.


doi:10.6342/NTU202301463

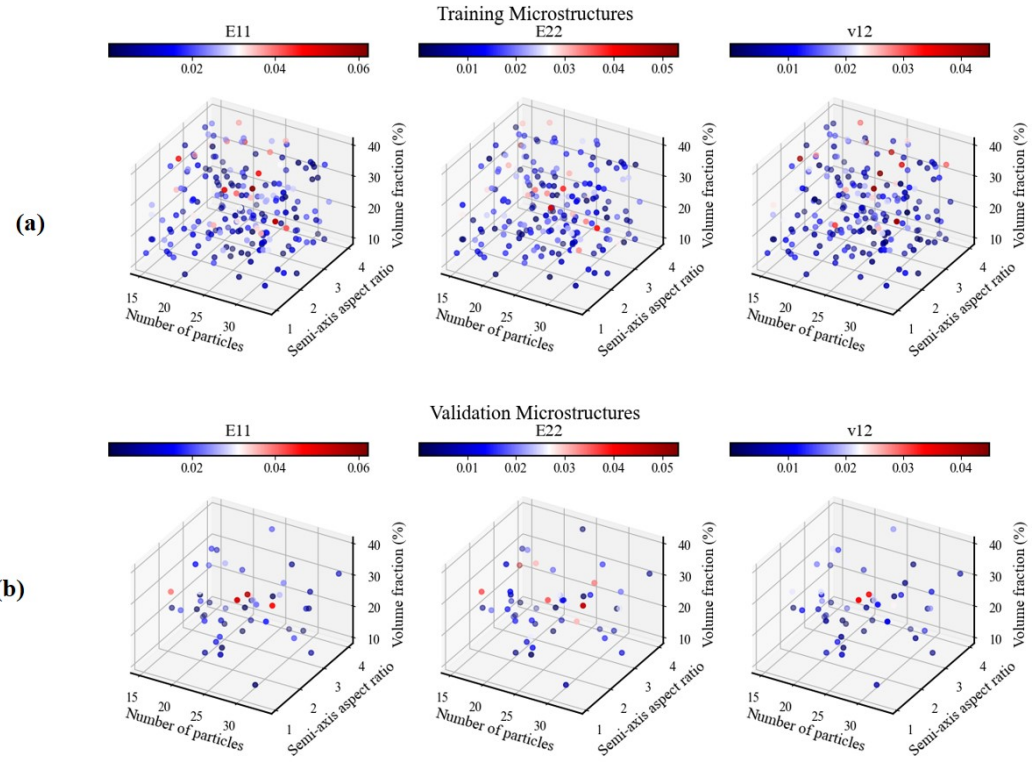



Figure 5.13: Relative errors on the predicted longitudinal Young's modulus $E_{11}$, transverse Young's modulus $E_{22}$ , and Poisson's ratio $v_{12}$ for microstructures in the (a) training set and (b) validation set.

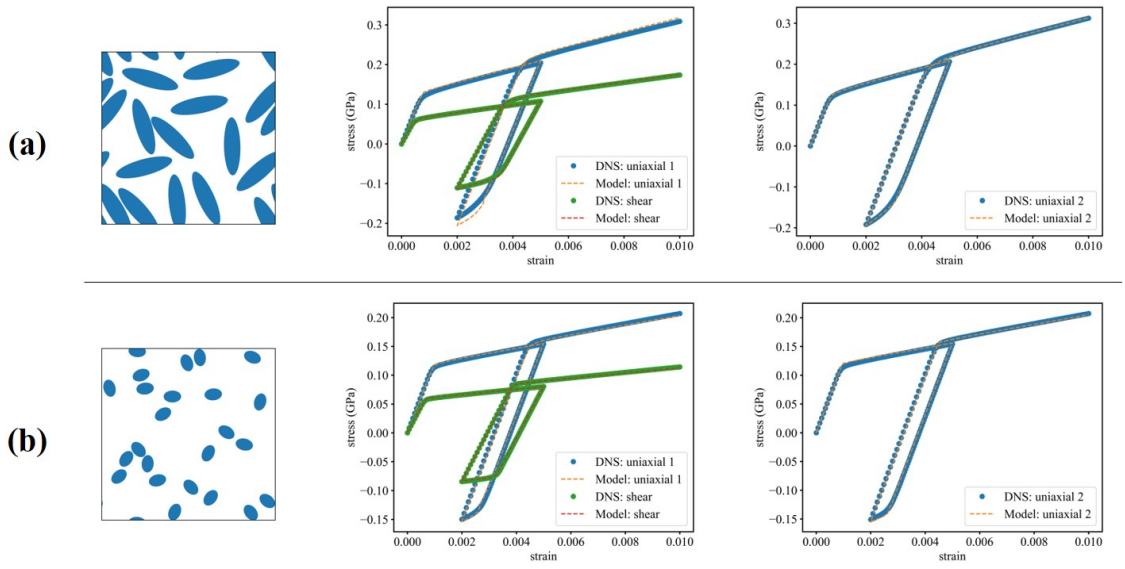



Figure 5.14: Nonlinear prediction with interpolated DMN models for selected microstructures in the training set. The properties used are listed in the Table 5.1



doi:10.6342/NTU202301463

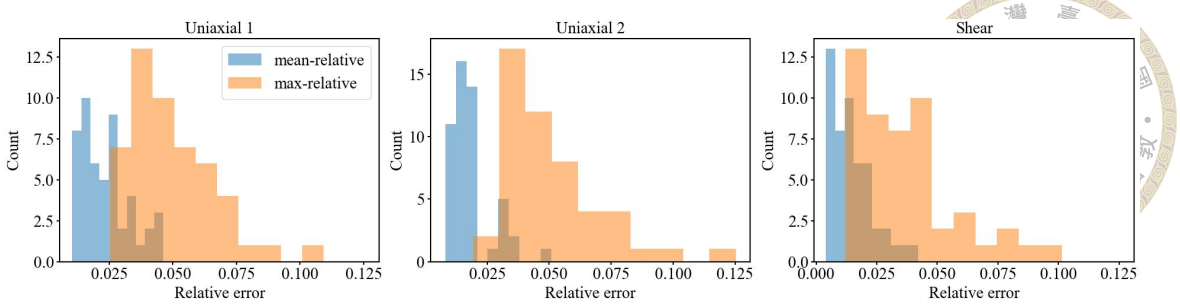



Figure 5.15: Distribution of the errors (for all the microstructures in the validation set) yielded by the generated DMN models for the nonlinear prediction performed based on the properties listed in Table 5.1.

We further assess these errors for all the microstructures unseen during training, i.e., in the validation set. The resulting histograms are displayed in Fig 5.15. Considering the mean-relative error, the average of the distributions are respectively 2.36%, 1.82%, and 1.50% for the three tests, while their maxima are 4.62%, 5.09%, and 4.22%. As to the max-relative error, the histograms average 4.93%, 5.06%, and 3.71%. They do not respectively exceed 10.93%, 12.53%, and 10.15%. These statistics are comparable with results report by Huang et al. [21]. Beyond the simple case considered in the first example, our proposed model shows its capacity to learn the topology of more complex microstructures and generate appropriate DMNs for them.


doi:10.6342/NTU202301463



doi:10.6342/NTU202301463

# Chapter 6 Conclusions and Future Work

In this work, we have explored the implementation of a graph-based model aimed at unifying the under a single umbrella the multiscale modeling of microstructures. Whereas the original formulation of DMN (eq. (3.18))

$$\bar{\boldsymbol{D}} = \mathcal{F}(\boldsymbol{D}^1, \boldsymbol{D}^2 | \, z_N^{k=1,2,\dots,2^N}, \theta_{h=0,1,\dots,N}^{k=1,2,\dots,2^h}) \tag{6.1}$$

focused on reducing the representation of a single RVE. This model further takes as input the graph representations of RVEs (eq. (4.22))

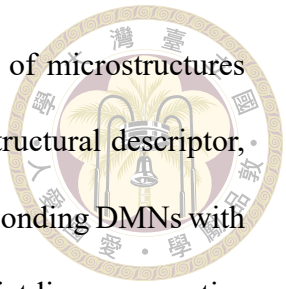
$$\bar{\boldsymbol{D}} = \mathcal{M}(\boldsymbol{G}, \boldsymbol{D}^1, \boldsymbol{D}^2 \mid \Theta_{\mathcal{H}}, \Theta_{\mathcal{T}}, \bar{\boldsymbol{p}}) \tag{6.2}$$

thus making it readily applicable to different microstructures. Besides, the model generates, as part of its pipeline, parameters for DMNs representing the input microstructures (eq. (4.20))

$$\tilde{\boldsymbol{p}} = \tilde{\boldsymbol{p}}(\boldsymbol{G} \mid \Theta_{\mathcal{H}}, \Theta_{\mathcal{T}}, \bar{\boldsymbol{p}}) \tag{6.3}$$

thus making it a tool for compressing RVEs into equivalent DMNs. Different examples showed the capability of the model to discriminate among dissimilar microstructures. In


doi:10.6342/NTU202301463



the first example, the model was successfully applied to a diversity of microstructures with fixed-size circular inclusions. These RVEs had a single microstructural descriptor, i.e., the inclusion volume fraction. The model learned to create corresponding DMNs with parameters reflecting this descriptor. The use of these DMNs to predict linear properties and nonlinear behavior was illustrated through several results. These capabilities were also demonstrated in a second example considering microstructures with multidimensional descriptors, and whose inclusions varied in shape, size, and orientation. The key features of our approach can be summarized as follows:

1. A model that captures detailed information about microstructures through their graph representation. It can potentially handle any type of complex microstructures, even those for which descriptors cannot be properly defined. Other works aimed at generalizing DMN [11, 21, 30] are limited in this respect since they are limited to microstructures with explicitly defined descriptors.

2. A model that is able to generate independent DMNs, even for unseen microstructures. The thus obtained DMNs are stand-alone models that can be used separately for online inference, thus leveraging the advantages of using a model with few degrees of freedom.

Our approach leaves open the door for the possibility of developing a large-scale model (in analogy with large language models), whose purpose is to generate twin DMNs for a diversity of microstructures. To achieve this goal, we envision 3D GNN-DMNs trained on various kinds of heterogeneous material systems. These would include particle-based, fiber-reinforced composites, polycrystalline materials, etc. Such an endeavor would help accelerate our understanding of materials. In reverse, this model is a perfect candi-


doi:10.6342/NTU202301463

date for the development of new materials, i.e. inverse material design. Given its graph input, it enables the possibility of designing materials at their microstructural level for specific properties. However, for this goal to be attained, extensive research needs to be undertaken to enhance the expressivity of the model in extracting graph features. Besides, it would greatly benefit from physically-based treatments to prevent its overfitting tendency.


doi:10.6342/NTU202301463



doi:10.6342/NTU202301463

# References


[1] S. G. Advani and C. L. Tucker III. The use of tensors to describe and predict fiber orientation in short fiber composites. Journal of rheology, 31(8):751–784, 1987.

[2] A. Agrawal and A. Choudhary. Perspective: Materials informatics and big data: Realization of the "fourth paradigm" of science in materials science. Apl Materials, 4(5):053208, 2016.

[3] F. M. Bianchi, D. Grattarola, and C. Alippi. Spectral clustering with graph neural networks for graph pooling. In International conference on machine learning, pages 874–883. PMLR, 2020.

[4] D. Bishara, Y. Xie, W. K. Liu, and S. Li. A state-of-the-art review on machine learning-based multiscale modeling, simulation, homogenization and design of materials. Archives of computational methods in engineering, 30(1):191–222, 2023.

[5] F. E. Bock, R. C. Aydin, C. J. Cyron, N. Huber, S. R. Kalidindi, and B. Klusemann. A review of the application of machine learning and data mining approaches in continuum materials mechanics. Frontiers in Materials, 6:110, 2019.

[6] S. Brody, U. Alon, and E. Yahav. How attentive are graph attention networks? arXiv preprint arXiv:2105.14491, 2021.




doi:10.6342/NTU202301463

[7] J. P. Crutchfield. The origins of computational mechanics: A brief intellectual history and several clarifications. arXiv preprint arXiv:1710.06832, 2017.

[8] W. M. Czarnecki, S. Osindero, M. Jaderberg, G. Swirszcz, and R. Pascanu. Sobolev training for neural networks. Advances in neural information processing systems, 30, 2017.

[9] B. Eidel. Deep cnns as universal predictors of elasticity tensors in homogenization. Computer Methods in Applied Mechanics and Engineering, 403:115741, 2023.

[10] S. Gajek, M. Schneider, and T. Böhlke. On the micromechanics of deep material networks. Journal of the Mechanics and Physics of Solids, 142:103984, 2020.

[11] S. Gajek, M. Schneider, and T. Böhlke. An fe–dmn method for the multiscale analysis of short fiber reinforced plastic components. Computer Methods in Applied Mechanics and Engineering, 384:113952, 2021.

[12] G. Galilei. Two new sciences. Dover, 1914.

[13] M. G. Geers, V. G. Kouznetsova, K. Matouš, and J. Yvonnet. Homogenization methods and multiscale modeling: nonlinear problems. Encyclopedia of computational mechanics second edition, pages 1–34, 2017.

[14] 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.

[15] K. Gupta and J. Meek. A brief history of the beginning of the finite element method. International journal for numerical methods in engineering, 39(22):3761–3774, 1996.



doi:10.6342/NTU202301463

[16] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. Advances in neural information processing systems, 30, 2017.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[18] R. Hill. Elastic properties of reinforced solids: some theoretical principles. Journal of the Mechanics and Physics of Solids, 11(5):357–372, 1963.

[19] R. Hill. Continuum micro-mechanics of elastoplastic polycrystals. Journal of the Mechanics and Physics of Solids, 13(2):89–101, 1965.

[20] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017.

[21] T. Huang, Z. Liu, C. Wu, and W. Chen. Microstructure-guided deep material network for rapid nonlinear material modeling and uncertainty quantification. Computer Methods in Applied Mechanics and Engineering, 398:115197, 2022.

[22] R. Jones, C. Safta, and A. Frankel. Deep learning and multi-level featurization of graph representations of microstructural data. Computational Mechanics, pages 1–19, 2023.

[23] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6):84–90, 2017.



doi:10.6342/NTU202301463

[25] M. Lefik, D. Boso, and B. Schrefler. Artificial neural networks in numerical modelling of composites. Computer Methods in Applied Mechanics and Engineering, 198(21-26):1785–1804, 2009.

[26] C. Liu, Y. Zhan, C. Li, B. Du, J. Wu, W. Hu, T. Liu, and D. Tao. Graph pooling for graph neural networks: Progress, challenges, and opportunities. arXiv preprint arXiv:2204.07321, 2022.

[27] X. Liu, S. Tian, F. Tao, and W. Yu. A review of artificial neural networks in the constitutive modeling of composite materials. Composites Part B: Engineering, 224:109152, 2021.

[28] Z. Liu and C. Wu. Exploring the 3d architectures of deep material network in data-driven multiscale mechanics. Journal of the Mechanics and Physics of Solids, 127:20–46, 2019.

[29] Z. Liu, C. Wu, and M. Koishi. A deep material network for multiscale topology learning and accelerated nonlinear modeling of heterogeneous materials. Computer Methods in Applied Mechanics and Engineering, 345:1138–1168, 2019.

[30] Z. Liu, C. Wu, and M. Koishi. Transfer learning of deep material network for seamless structure–property predictions. Computational Mechanics, 64(2):451–465, 2019.

[31] Z. Liu, C. T. Wu, B. Ren, W. K. Liu, and R. G. Grimes. Multiscale simulations of material with heterogeneous structures based on representative volume element techniques. 2018.

[32] G. Lu and E. Kaxiras. An overview of multiscale simulations of materials. arXiv preprint cond-mat/0401073, 2004.



doi:10.6342/NTU202301463

[33] J.-C. Michel, H. Moulinec, and P. Suquet. Effective properties of composite materials with periodic microstructure: a computational approach. Computer methods in applied mechanics and engineering, 172(1-4):109–143, 1999.

[34] 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.

[35] M. Mozaffar, R. Bostanabad, W. Chen, K. Ehmann, J. Cao, and M. Bessa. Deep learning predicts path-dependent plasticity. Proceedings of the National Academy of Sciences, 116(52):26414–26420, 2019.

[36] I. Newton. Philosophiae naturalis principia mathematica, volume 1. G. Brookman, 1833.

[37] L. Noels et al. Micromechanics-based material networks revisited from the interaction viewpoint; robust and efficient implementation for multi-phase composites. European Journal of Mechanics-A/Solids, 91:104384, 2022.

[38] C. Rao and Y. Liu. Three-dimensional convolutional neural network (3d-cnn) for heterogeneous material homogenization. Computational Materials Science, 184:109850, 2020.

[39] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[40] T.-H. Su, S.-J. Huang, J. G. Jean, and C.-S. Chen. Multiscale computational solid mechanics: data and machine learning. Journal of Mechanics, 38:568–585, 2022.

[41] P. Suquet. Elements of homogenization for inelastic solid mechanics. Homogenization techniques for composite media, 1987.



doi:10.6342/NTU202301463

[42] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In Proceedings of the AAAI conference on artificial intelligence, volume 31, 2017.

[43] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1–9, 2015.

[44] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. arXiv preprint arXiv:1710.10903, 2017.

[45] N. N. Vlassis, R. Ma, and W. Sun. Geometric deep learning for computational mechanics part i: anisotropic hyperelasticity. Computer Methods in Applied Mechanics and Engineering, 371:113299, 2020.

[46] N. N. Vlassis and W. Sun. Component-based machine learning paradigm for discovering rate-dependent and pressure-sensitive level-set plasticity models. Journal of Applied Mechanics, 89(2), 2022.

[47] N. N. Vlassis and W. Sun. Geometric deep learning for computational mechanics part ii: Graph embedding for interpretable multiscale plasticity. arXiv preprint arXiv:2208.00246, 2022.

[48] L. Wu, L. Adam, and L. Noels. Micro-mechanics and data-driven based reduced order models for multi-scale analyses of woven composites. Composite Structures, 270:114058, 2021.

[49] L. Wu, N. G. Kilingar, L. Noels, et al. A recurrent neural network-accelerated multi-scale model for elasto-plastic heterogeneous materials subjected to random cyclic



doi:10.6342/NTU202301463

and non-proportional loading paths. Computer Methods in Applied Mechanics and Engineering, 369:113234, 2020.

[50] C. Yang, Y. Kim, S. Ryu, and G. X. Gu. Prediction of composite microstructure stress-strain curves using convolutional neural networks. Materials & Design, 189:108509, 2020.

[51] H. Yang, H. Qiu, Q. Xiang, S. Tang, and X. Guo. Exploring elastoplastic constitutive law of microstructured materials through artificial neural network—a mechanistic-based data-driven approach. Journal of Applied Mechanics, 87(9):091005, 2020.

[52] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec. Hierarchical graph representation learning with differentiable pooling. Advances in neural information processing systems, 31, 2018.

[53] J. Yvonnet. Computational homogenization of heterogeneous materials with finite elements, volume 258. Springer, 2019.



doi:10.6342/NTU202301463



doi:10.6342/NTU202301463

# Appendix A — Derivations

## A.1 Localization functions

### A.1.1 Derivation of Stress Concentration Matrix

Expanding the equality in equation 3.4, we obtain:

$$D_{11}^1\sigma_1^1 + [D_{12}^1\sigma_2^1 + D_{13}^1\sigma_3^1] = D_{11}^2\sigma_1^2 + [D_{12}^2\sigma_2^2 + D_{13}^2\sigma_3^2] \tag{A.4}$$

From the averaging relation in equation 3.8, we may write $\boldsymbol{\sigma}^2$ as:

$$\boldsymbol{\sigma}^2 = \frac{1}{f_2}\boldsymbol{\sigma} - \frac{f_1}{f_2}\boldsymbol{\sigma}^1$$

By then substituting this relationship in equation A.11, we obtain:

$$D_{11}^1\sigma_1^1 + [D_{12}^1\sigma_2^1 + D_{13}^1\sigma_3^1] = D_{11}^2(\frac{1}{f_2}\sigma_1 - \frac{f_1}{f_2}\sigma_1^1) + [D_{12}^2\sigma_2^2 + D_{13}^2\sigma_3^2] \tag{A.5}$$

After multiplying both sides of equation A.12 by $f_2$, and re-arranging, we may write:

$$(f_2D_{11}^1 + f_1D_{11}^2)\sigma_1^1 = D_{11}^2\sigma_1 + f_2[D_{12}^2\sigma_2^2 + D_{13}^2\sigma_3^2] - f_2[D_{12}^1\sigma_2^1 + D_{13}^1\sigma_3^1] \tag{A.6}$$



doi:10.6342/NTU202301463

But, we have from the equilibrium conditions in equation 3.9:

$$\sigma_2^1 = \sigma_2^2 = \sigma_2$$

$$\sigma_3^1 = \sigma_3^2 = \sigma_3$$

Equation A.13 then becomes:

$$(f_2 D_{11}^1 + f_1 D_{11}^2)\sigma_1^1 = D_{11}^2\sigma_1 + f_2[(D_{12}^2 - D_{12}^1)\sigma_2 + (D_{13}^2 - D_{13}^1)\sigma_3] \tag{A.7}$$

With $\Delta \boldsymbol{D} := \boldsymbol{D}^2 - \boldsymbol{D}^1$, equation A.14 may be modified as:

$$(f_2 D_{11}^1 + f_1 D_{11}^2)\sigma_1^1 = D_{11}^2\sigma_1 + f_2[\Delta D_{12}\sigma_2 + \Delta D_{13}\sigma_3] \tag{A.8}$$

Let $\gamma = f_2 D_{11}^1 + f_1 D_{11}^2$, equation A.15 may be used to show that $\sigma_1^1$ can be computed from the homogenized stresses through the relationship:

$$\sigma_1^1 = \frac{1}{\gamma}\begin{bmatrix} D_{11}^2 & f_2\Delta D_{12} & f_2\Delta D_{13} \end{bmatrix}\begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \end{bmatrix} \tag{A.9}$$

Let's denote the stress concentration matrix $\boldsymbol{b}^1$ relating macroscopic and microscopic stresses such that $\boldsymbol{\sigma}^1 = \boldsymbol{b}^1\boldsymbol{\sigma}$, then

$$\boldsymbol{b}^1 := \begin{bmatrix} D_{11}^2/\gamma & f_2\Delta D_{12}/\gamma & f_2\Delta D_{13}/\gamma \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.10}$$



doi:10.6342/NTU202301463

The stress concentration matrix $\boldsymbol{b}^2$ relating $\boldsymbol{\sigma}^2 = \boldsymbol{b}^2\boldsymbol{\sigma}$ can also be derived by following a similar procedure.

### A.1.2 Derivation of Strain Concentration Matrix

Expanding the equilibrium condition in equation 3.9, we obtain:

$$\begin{bmatrix} C_{22}^1 & C_{23}^1 \\ C_{23}^1 & C_{33}^1 \end{bmatrix} \begin{bmatrix} \epsilon_2^1 \\ \epsilon_3^1 \end{bmatrix} + \begin{bmatrix} C_{12}^1\epsilon_1^1 \\ C_{13}^1\epsilon_1^1 \end{bmatrix} = \begin{bmatrix} C_{22}^2 & C_{23}^2 \\ C_{23}^2 & C_{33}^2 \end{bmatrix} \begin{bmatrix} \epsilon_2^2 \\ \epsilon_3^2 \end{bmatrix} + \begin{bmatrix} C_{12}^2\epsilon_1^2 \\ C_{13}^2\epsilon_1^2 \end{bmatrix} \tag{A.11}$$

From the strain averaging equation in 3.7, we may write $\boldsymbol{\epsilon}^2$ as:

$$\boldsymbol{\epsilon}^2 = \frac{1}{f_2}\boldsymbol{\epsilon} - \frac{f_1}{f_2}\boldsymbol{\epsilon}^1$$

By then substituting this relationship in A.11, we obtain:

$$\begin{bmatrix} C_{22}^1 & C_{23}^1 \\ C_{23}^1 & C_{33}^1 \end{bmatrix} \begin{bmatrix} \epsilon_2^1 \\ \epsilon_3^1 \end{bmatrix} + \begin{bmatrix} C_{12}^1\epsilon_1^1 \\ C_{13}^1\epsilon_1^1 \end{bmatrix} = \begin{bmatrix} C_{22}^2 & C_{23}^2 \\ C_{23}^2 & C_{33}^2 \end{bmatrix} \left(\frac{1}{f_2}\begin{bmatrix} \epsilon_2 \\ \epsilon_3 \end{bmatrix} - \frac{f_1}{f_2}\begin{bmatrix} \epsilon_2^1 \\ \epsilon_3^1 \end{bmatrix}\right) + \begin{bmatrix} C_{12}^2\epsilon_1^2 \\ C_{13}^2\epsilon_1^2 \end{bmatrix} \tag{A.12}$$

After multiplying both sides of equation A.12 by $f_2$ and re-arranging, we may write:

$$\left(f_2\begin{bmatrix} C_{22}^1 & C_{23}^1 \\ C_{23}^1 & C_{33}^1 \end{bmatrix} + f_1\begin{bmatrix} C_{22}^2 & C_{23}^2 \\ C_{23}^2 & C_{33}^2 \end{bmatrix}\right)\begin{bmatrix} \epsilon_2^1 \\ \epsilon_3^1 \end{bmatrix} = \begin{bmatrix} C_{22}^2 & C_{23}^2 \\ C_{23}^2 & C_{33}^2 \end{bmatrix}\begin{bmatrix} \epsilon_2 \\ \epsilon_3 \end{bmatrix} + f_2\left(\begin{bmatrix} C_{12}^2\epsilon_1^2 \\ C_{13}^2\epsilon_1^2 \end{bmatrix} - \begin{bmatrix} C_{12}^1\epsilon_1^1 \\ C_{13}^1\epsilon_1^1 \end{bmatrix}\right) \tag{A.13}$$

But, we have from the kinematic equilibrium: $\epsilon_1^1 = \epsilon_1^2 = \epsilon_1$. Equation A.13 then becomes:

$$\left(f_2\begin{bmatrix} C_{22}^1 & C_{23}^1 \\ C_{23}^1 & C_{33}^1 \end{bmatrix} + f_1\begin{bmatrix} C_{22}^2 & C_{23}^2 \\ C_{23}^2 & C_{33}^2 \end{bmatrix}\right)\begin{bmatrix} \epsilon_2^1 \\ \epsilon_3^1 \end{bmatrix} = \begin{bmatrix} C_{22}^2 & C_{23}^2 \\ C_{23}^2 & C_{33}^2 \end{bmatrix}\begin{bmatrix} \epsilon_2 \\ \epsilon_3 \end{bmatrix} + f_2\begin{bmatrix} (C_{12}^2 - C_{12}^1)\epsilon_1 \\ (C_{13}^2 - C_{13}^1)\epsilon_1 \end{bmatrix} \tag{A.14}$$

With $\Delta\boldsymbol{C} := \boldsymbol{C}^2 - \boldsymbol{C}^1$ and $\hat{\boldsymbol{C}} = f_2\boldsymbol{C}^1 + f_1\boldsymbol{C}^2$, equation A.14 may be modified as:


doi:10.6342/NTU202301463

$$\begin{bmatrix} \hat{C}^1_{22} & \hat{C}^1_{23} \\ \hat{C}^1_{23} & \hat{C}^1_{33} \end{bmatrix} \begin{bmatrix} \epsilon^1_2 \\ \epsilon^1_3 \end{bmatrix} = \begin{bmatrix} C^2_{22} & C^2_{23} \\ C^2_{23} & C^2_{33} \end{bmatrix} \begin{bmatrix} \epsilon_2 \\ \epsilon_3 \end{bmatrix} + f_2 \begin{bmatrix} \Delta C_{12}\epsilon_1 \\ \Delta C_{13}\epsilon_1 \end{bmatrix} \tag{A.15}$$

By re-arranging equation A.15, it is shown that $[\epsilon^1_2\ \epsilon^1_3]^T$ can be computed from the homogenized strains through the relationship:

$$\begin{bmatrix} \epsilon^1_2 \\ \epsilon^1_3 \end{bmatrix} = \begin{bmatrix} \hat{C}^1_{22} & \hat{C}^1_{23} \\ \hat{C}^1_{23} & \hat{C}^1_{33} \end{bmatrix}^{-1} \begin{bmatrix} f_2\Delta C_{12} & C^2_{22} & C^2_{23} \\ f_2\Delta C_{13} & C^2_{23} & C^2_{33} \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} \tag{A.16}$$

We define $\boldsymbol{s}_{23}$ as:

$$\mathbf{s}_{23} = \begin{bmatrix} \hat{C}^1_{22} & \hat{C}^1_{23} \\ \hat{C}^1_{23} & \hat{C}^1_{33} \end{bmatrix}^{-1} \begin{bmatrix} f_2\Delta C_{12} & C^2_{22} & C^2_{23} \\ f_2\Delta C_{13} & C^2_{23} & C^2_{33} \end{bmatrix} \tag{A.17}$$

Let's denote the strain concentration matrix $\boldsymbol{s}^1$ relating macroscopic and microscopic strains such that $\boldsymbol{\epsilon}^1 = \boldsymbol{s}^1\boldsymbol{\epsilon}$ , then

$$\boldsymbol{s}^1 := \begin{bmatrix} 1 & 0 & 0 \\ & \boldsymbol{s}_{23} & \end{bmatrix} \tag{A.18}$$

The strain concentration matrix $\boldsymbol{s}^2$ relating $\boldsymbol{\epsilon}^2 = \boldsymbol{s}^2\boldsymbol{\epsilon}$ can also be derived by following a similar procedure.



doi:10.6342/NTU202301463

## A.2 Homogenization functions

### A.2.1 Homogenized Compliance Matrix

The strain averaging equation 3.7 may also be written as:

$$\begin{aligned}
\boldsymbol{\epsilon} &= f_1 \boldsymbol{D}^1 \boldsymbol{\sigma}^1 + f_2 \boldsymbol{D}^2 \boldsymbol{\sigma}^2 \\
&= f_1 \boldsymbol{D}^1 \boldsymbol{\sigma}^1 + f_2 \boldsymbol{D}^2 (\frac{1}{f_2}\boldsymbol{\sigma} - \frac{f_1}{f_2}\boldsymbol{\sigma}^1) \\
&= \boldsymbol{D}^2 \boldsymbol{\sigma} - f_1 (\boldsymbol{D}^2 - \boldsymbol{D}^1) \boldsymbol{\sigma}^1 \\
&= \boldsymbol{D}^2 \boldsymbol{\sigma} - f_1 (\boldsymbol{D}^2 - \boldsymbol{D}^1) \boldsymbol{b}^1 \boldsymbol{\sigma} \\
&= (\boldsymbol{D}^2 - f_1 \Delta \boldsymbol{D} \boldsymbol{b}^1) \boldsymbol{\sigma}
\end{aligned} \tag{A.19}$$

The homogenized compliance matrix $\bar{\boldsymbol{D}}^r$ such that $\boldsymbol{\epsilon} = \bar{\boldsymbol{D}}^r \boldsymbol{\sigma}$ is therefore:

$$\bar{\boldsymbol{D}}^r \equiv \boldsymbol{D}^2 - f_1 \Delta \boldsymbol{D} \boldsymbol{b}^1 \tag{A.20}$$

### A.2.2 Homogenized Stiffness Matrix

The stress averaging equation 3.8 may be written as:

$$\begin{aligned}
\boldsymbol{\sigma} &= f_1 \boldsymbol{\sigma}^1 + f_2 \boldsymbol{\sigma}^2 \\
&= f_1 \boldsymbol{C}^1 \boldsymbol{\epsilon}^1 + f_2 \boldsymbol{C}^2 \boldsymbol{\epsilon}^2 \\
&= f_1 \boldsymbol{C}^1 \boldsymbol{\epsilon}^1 + f_2 \boldsymbol{C}^2 (\frac{1}{f_2}\boldsymbol{\epsilon} - \frac{f_1}{f_2}\boldsymbol{\epsilon}^1) \\
&= \boldsymbol{C}^2 \boldsymbol{\epsilon} - f_1 (\boldsymbol{C}^2 - \boldsymbol{C}^1) \boldsymbol{\epsilon}^1 \\
&= \boldsymbol{C}^2 \boldsymbol{\epsilon} - f_1 (\boldsymbol{C}^2 - \boldsymbol{C}^1) \boldsymbol{s}^1 \boldsymbol{\epsilon} \\
&= (\boldsymbol{C}^2 - f_1 \Delta \boldsymbol{C} \boldsymbol{s}^1) \boldsymbol{\epsilon}
\end{aligned} \tag{A.21}$$



doi:10.6342/NTU202301463

The homogenized stiffness matrix $\bar{\boldsymbol{C}}^r$ such that $\boldsymbol{\sigma} = \bar{\boldsymbol{C}}^r \boldsymbol{\epsilon}$ is therefore:

$$\bar{\boldsymbol{C}}^r \equiv \boldsymbol{C}^2 - f_1 \Delta \boldsymbol{C} \boldsymbol{s}^1 \tag{A.22}$$



doi:10.6342/NTU202301463

# Appendix B — Algorithms

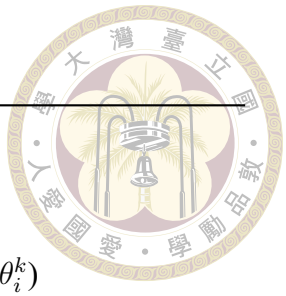
## B.1 Nonlinear Prediction

**Algorithm 1** Online Prediction in DMN

**procedure** ONLINE PREDICTION
▷ $\Delta\bar{\boldsymbol{\epsilon}}$: applied macroscopic strain at load step $n$
▷ $n$: load step
▷ $s$: iteration within load step $n$
▷ $N$: number of layers in DMN
▷ ${}_{n-1}\boldsymbol{B}_N^j$: history variables from the last load step
▷ TOL: tolerance for convergence

▷ **0. Initialization**
**if** $s = 1$ **then**
▷ Initialize incremental strains and stresses in network
$\Delta\boldsymbol{\sigma}_{i,s-1}^k \leftarrow \mathbf{0}$
$\Delta\boldsymbol{\epsilon}_{i,s-1}^k \leftarrow \mathbf{0}$
▷ Initialize residual strains in bottom layer $N$
$\delta\boldsymbol{\epsilon}_{N,s}^k \leftarrow \mathbf{0}$
**end if**


doi:10.6342/NTU202301463



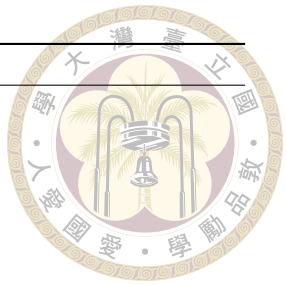
▷ **1. Upscaling**
**for** $i = N-1, N-2, \ldots, 0$ **do**
▷ Upscale tangents (Alg. 2)
$\boldsymbol{D}^k_{i,s} \leftarrow$ UPSCALECOMPLIANCE($\boldsymbol{D}^{2k-1}_{i+1,s}, \boldsymbol{D}^{2k}_{i+1,s}, f^{2k-1}_{i+1}, f^{2k}_{i+1}, \theta^k_i$)
▷ Upscale residual strains (Alg. 3)
$\delta\boldsymbol{\epsilon}^k_{i,s} \leftarrow$ UPSCALESTRAIN($\delta\boldsymbol{\epsilon}^{2k-1}_{i+1,s}, \delta\boldsymbol{\epsilon}^{2k}_{i+1,s}, \boldsymbol{D}^{2k-1}_{i+1,s}, \boldsymbol{D}^{2k}_{i+1,s}, f^{2k-1}_{i+1}, f^{2k}_{i+1}, \theta^k_i$)
**end for**

▷ **2. Evaluate boundary conditions**
$\boldsymbol{x} \leftarrow \text{inv}(\boldsymbol{D}_{0,s})(\Delta\boldsymbol{\epsilon}_{0,s-1} - \Delta\bar{\boldsymbol{\epsilon}})$ ▷ Netwon correction term
$\Delta\boldsymbol{\sigma}_{0,s} \leftarrow \Delta\boldsymbol{\sigma}_{0,s-1} - \boldsymbol{x}$
$\Delta\boldsymbol{\epsilon}_{0,s} \leftarrow \boldsymbol{D}_{0,s}\Delta\boldsymbol{\sigma}_{0,s} + \delta\boldsymbol{\epsilon}_{0,s}$

▷ **3. Downscaling and Evaluation of constitutive laws**
**for** $i = 0, 1, \ldots, N$ **do**
**if** $i \neq N$ **then**
▷ Downscale strain/stress (Alg. 4 and 5)
$\Delta\boldsymbol{\epsilon}^{2k}_{i+1,s}, \Delta\boldsymbol{\epsilon}^{2k-1}_{i+1,s} \leftarrow$ DOWNSCALESTRAIN($\Delta\boldsymbol{\epsilon}^k_{i,s}, \theta^k_{i,s}, \boldsymbol{D}^{2k-1}_{i+1,s}, \boldsymbol{D}^{2k}_{i+1,s}, f^{2k-1}_{i+1}, f^{2k}_{i+1}$)
$\Delta\boldsymbol{\sigma}^{2k}_{i+1,s}, \Delta\boldsymbol{\sigma}^{2k-1}_{i+1,s} \leftarrow$ DOWNSCALESTRESS($\Delta\boldsymbol{\sigma}^k_{i,s}, \theta^k_{i,s}, \boldsymbol{D}^{2k-1}_{i+1,s}, \boldsymbol{D}^{2k}_{i+1,s}, f^{2k-1}_{i+1}, f^{2k}_{i+1}$)
**else if** $i = N$ **then**
▷ 1. Undo rotation
$\alpha \leftarrow -\theta^k_{N,s}$
$\Delta\boldsymbol{\epsilon}^k_{N,s} \leftarrow$ ROTATESTATE($\Delta\boldsymbol{\epsilon}^k_{N,s}, \alpha$) ▷ Alg. 6
▷ 2. Evaluate material laws
**for** $j = 1, \ldots, 2^N$ **do**
▷ Calculate change in incremental strain
$err^j_N = ||\Delta\boldsymbol{\epsilon}^j_{N,s} - \Delta\boldsymbol{\epsilon}^j_{N,s-1}||$
▷ Evaluate constitutive material laws by Alg. 7
$\Delta\boldsymbol{\sigma}^j_{N,s+1}, \boldsymbol{D}^j_{N,s+1}, \delta\boldsymbol{\epsilon}^j_{N,s+1}, \boldsymbol{\beta}^j_N \leftarrow$ MATLAW($\Delta\boldsymbol{\epsilon}^j_{N,s}, \boldsymbol{\epsilon}^j_{N,s}, \boldsymbol{\sigma}^j_N, {}_{n-1}\boldsymbol{B}^j_N$)
**end for**
**end if**
**end for**

▷ **4. Convergence Check**
**if** $err^j_N < \text{TOL}, \forall j, N$ **then**
▷ all bottom nodes have converged
${}_n\boldsymbol{B}^j_N \leftarrow \boldsymbol{\beta}^j_N$
$n \leftarrow n + 1$
Go to **0. Initialization**
**else**
$s \leftarrow s + 1$
Go to **1. Upscaling**
**end if**
**end procedure**


doi:10.6342/NTU202301463



**Algorithm 2** Homogenization of compliance matrices

**function** UPSCALECOMPLIANCE($\boldsymbol{D}^1, \boldsymbol{D}^2, f_1, f_2, \theta$)
  ▷ $\boldsymbol{D}^1, \boldsymbol{D}^2$: compliance matrices at children nodes
  ▷ $f_1, f_2$: volume fractions of children nodes
  ▷ $\theta$: rotation angle

  ▷ Homogenize
  $\bar{\boldsymbol{D}}^r \leftarrow \boldsymbol{D}^2 - f_1 \Delta \boldsymbol{D} \boldsymbol{b}^1$ ▷ Equation (A.20)
  ▷ Rotate strain
  $\bar{\boldsymbol{D}} \leftarrow \boldsymbol{R}(-\theta) \bar{\boldsymbol{D}}^r \boldsymbol{R}(\theta)$ ▷ Equation (3.13)
  **return** $\bar{\boldsymbol{D}}$
**end function**

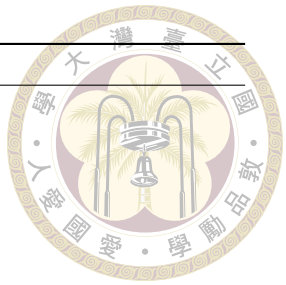
**Algorithm 3** Homogenization of residual strains

**function** UPSCALESTRAIN($\delta\boldsymbol{\epsilon}^1, \delta\boldsymbol{\epsilon}^2, \boldsymbol{D}^1, \boldsymbol{D}^2, f_1, f_2, \theta$)
  ▷ $\delta\boldsymbol{\epsilon}^1, \delta\boldsymbol{\epsilon}^2$: residual strains at children nodes
  ▷ $\boldsymbol{D}^1, \boldsymbol{D}^2$: compliance matrices from children nodes
  ▷ $f_1, f_2$: volume fractions of children nodes
  ▷ $\theta$: rotation angle

  $\Gamma := f_2 D^1_{11} + f_1 D^2_{11}$
  $\delta\bar{\epsilon}^r_{11} = \frac{1}{\Gamma}(f_1 D^2_{11} \delta\epsilon^1_{11} + f_2 D^1_{11} \delta\epsilon^2_{11})$
  $\delta\bar{\epsilon}^r_{22} = f_1 \delta\epsilon^1_{22} + f_2 \delta\epsilon^2_{22} - \frac{1}{\Gamma} f_1 f_2 (D^1_{12} - D^2_{12})(\delta\epsilon^1_{11} - \delta\epsilon^2_{11})$
  $\delta\bar{\epsilon}^r_{12} = f_1 \delta\epsilon^1_{12} + f_2 \delta\epsilon^2_{12} - \frac{1}{\Gamma} f_1 f_2 (D^1_{13} - D^2_{13})(\delta\epsilon^1_{11} - \delta\epsilon^2_{11})$

  ▷ Rotate strain
  $\delta\bar{\boldsymbol{\epsilon}} \leftarrow$ ROTATESTATE($\delta\bar{\boldsymbol{\epsilon}}^r, \theta$)
  **return** $\delta\bar{\boldsymbol{\epsilon}}$
**end function**


doi:10.6342/NTU202301463

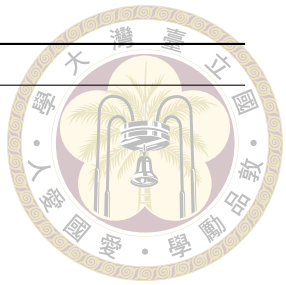


**Algorithm 4** Downscaling of incremental strain

**function** DOWNSCALESTRAIN($\Delta\bar{\boldsymbol{\epsilon}}, \theta, \boldsymbol{D}^1, \boldsymbol{D}^2, f_1, f_2$)
    ▷ $\Delta\bar{\boldsymbol{\epsilon}}$: incremental strain at parent node
    ▷ $\theta$: rotation angle at parent node
    ▷ $\boldsymbol{D}^1, \boldsymbol{D}^2$: compliance matrices of children nodes
    ▷ $f_1, f_2$: volume fractions of children nodes

    ▷ 1. Undo rotation
    $\alpha \leftarrow -\theta$
    $\Delta\bar{\boldsymbol{\epsilon}} \leftarrow$ ROTATESTATE($\Delta\bar{\boldsymbol{\epsilon}}, \alpha$)
    ▷ 2. Compute strain concentration matrices
    $\boldsymbol{s}^1, \boldsymbol{s}^2 \leftarrow$ `Compute from` $(\boldsymbol{D}^1, \boldsymbol{D}^2, f_1, f_2)$
    ▷ 3. Calculate strain at children nodes
    $\Delta\boldsymbol{\epsilon}^1 \leftarrow s^1 \Delta\bar{\boldsymbol{\epsilon}}$
    $\Delta\boldsymbol{\epsilon}^2 \leftarrow s^2 \Delta\bar{\boldsymbol{\epsilon}}$
    ▷ Exit
    **return** $\Delta\boldsymbol{\epsilon}^1, \Delta\boldsymbol{\epsilon}^2$
**end function**


doi:10.6342/NTU202301463



**Algorithm 5** Downscaling of incremental stress

**function** DOWNSCALESTRESS($\Delta\bar{\boldsymbol{\sigma}}, \theta, \boldsymbol{D}^1, \boldsymbol{D}^2, f_1, f_2$)
  ▷ $\Delta\bar{\boldsymbol{\sigma}}$: incremental stress at parent node
  ▷ $\theta$: rotation angle at parent node
  ▷ $\boldsymbol{D}^1, \boldsymbol{D}^2$: compliance matrices of children nodes
  ▷ $f_1, f_2$: volume fractions of children nodes

  ▷ 1. Undo rotation
  $\alpha \leftarrow -\theta$
  $\Delta\bar{\boldsymbol{\sigma}} \leftarrow$ ROTATESTATE($\Delta\bar{\boldsymbol{\sigma}}, \alpha$)
  ▷ 2. Compute stress concentration matrices
  $\boldsymbol{b}^1, \boldsymbol{b}^2 \leftarrow$ `Compute from` $(\boldsymbol{D}^1, \boldsymbol{D}^2, f_1, f_2)$
  ▷ 3. Calculate strain at children nodes
  $\Delta\boldsymbol{\sigma}^1 \leftarrow b^1\Delta\bar{\boldsymbol{\sigma}}$
  $\Delta\boldsymbol{\sigma}^2 \leftarrow b^2\Delta\bar{\boldsymbol{\sigma}}$
  ▷ Exit
  **return** $\Delta\boldsymbol{\sigma}^1, \Delta\boldsymbol{\sigma}^2$
**end function**

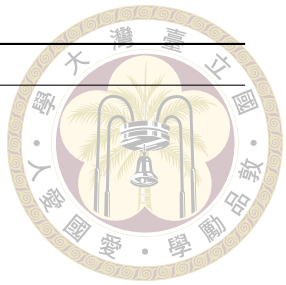
**Algorithm 6** Rotation of strain/stress

**function** ROTATESTATE($\boldsymbol{x}, \theta$)
  ▷ $\boldsymbol{x}$: strain/stress to be rotated
  ▷ $\theta$: rotation angles

$$\boldsymbol{R}(a) := \begin{bmatrix} \cos^2 a & \sin^2 a & \sqrt{2}\sin a\cos a \\ \sin^2 a & \cos^2 a & -\sqrt{2}\sin a\cos a \\ -\sqrt{2}\sin a\cos a & \sqrt{2}\sin a\cos a & \cos^2 a - \sin^2 a \end{bmatrix}$$

  $\boldsymbol{x} \leftarrow \boldsymbol{R}(-\theta)\boldsymbol{x}$ ▷ Rotation
  **return** $\boldsymbol{x}$
**end function**


doi:10.6342/NTU202301463



**Algorithm 7** Evaluate material law

**function** MatLaw($\Delta\boldsymbol{\epsilon}, \boldsymbol{\epsilon}, \boldsymbol{\sigma}, \boldsymbol{\beta}$)

▷ $\Delta\boldsymbol{\epsilon}$: incremental strain

▷ $\boldsymbol{\epsilon}$: total strain

▷ $\boldsymbol{\sigma}$: total stress

▷ $\boldsymbol{\beta}$: history variables

▷ Evaluate constitutive material laws

$\Delta\boldsymbol{\sigma} = \Delta\boldsymbol{\sigma}(\Delta\boldsymbol{\epsilon}, \boldsymbol{\epsilon}, \boldsymbol{\sigma}, \boldsymbol{\beta})$

$\boldsymbol{D} = \boldsymbol{D}(\Delta\boldsymbol{\epsilon}, \boldsymbol{\epsilon}, \boldsymbol{\sigma}, \boldsymbol{\beta})$

$\boldsymbol{\beta} = \boldsymbol{\beta}(\Delta\boldsymbol{\epsilon}, \boldsymbol{\epsilon}, \boldsymbol{\sigma}, \boldsymbol{\beta})$

▷ Calculate new residual strain

$\delta\boldsymbol{\epsilon} = \Delta\boldsymbol{\epsilon} - \boldsymbol{D}\Delta\boldsymbol{\sigma}$

▷ Return

**return** $\Delta\boldsymbol{\sigma}, \boldsymbol{D}, \delta\boldsymbol{\epsilon}, \boldsymbol{\beta}$

**end function**



doi:10.6342/NTU202301463

## B.2 Microstructure Generation

**Algorithm 8** Micro-structure creation

**function** CREATERVE($descr, sz$)
    ▷ $descr$: set of RVE geometric descriptors
    ▷ $sz$: size of RVE

    ▷ Initialization
    $RVE \leftarrow$ `<Create RVE of size` $(sz, sz)$`>`
    ▷ Sample particles
    $P \leftarrow$ `<Sample particles based on` $descr$ `and RVE size>`
    ▷ Distribute particles in RVE
    **for all** $p \in P$ **do**
        PLACEPARTICLE($p, RVE$)
    **end for**
**end function**


doi:10.6342/NTU202301463

```
14:
15: function PlaceParticle(p, RVE)
16:     ▷ p: particle
17:     ▷ RVE: RVE in which to place p
18:
19:     ovlp = False                    ▷ flag for whether particle overlaps another one
20:     crossBorder = False             ▷ flag for whether particle intersects RVE border
21:     while True do
22:         (x, y, θ) ← <Sample location and orientation of p>
23:         ▷ Check for overlapping
24:         ovlp ← <Evaluate if overlapping occurs>
25:         if ovlp then
26:             <Remove particle p>
27:             continue                ▷ re-sample location and orientation of particle p
28:         end if
29:         ▷ Ensure periodicity of RVE
30:         crossBorder ← <Evaluate if intersection with border occurs>
31:         if crossBorder then
32:             <Ensure periodicity of particle p>
33:             ovlp ← <Re-evaluate if overlapping occurs>
34:             if ovlp then
35:                 <Remove particle p>
36:                 continue            ▷ re-sample location and orientation of particle p
37:             else
38:                 return                          ▷ placement of particle p successful
39:             end if
40:         else
41:             return                              ▷ placement of particle p successful
42:         end if
43:     end while
44: end function
```



doi:10.6342/NTU202301463