# Investigating the performance of Deep Material Networks in accelerating multiscale modelling of laminated composites

by

J. J. Metz

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Thursday October 16th, 2020 at 11:46h.

| | | |
|---|---|---|
| Student number: | 4051017 | |
| Project duration: | September 1st, 2019 – October 16th, 2020 | |
| Thesis committee: | Dr. ir. F.P. (Frans) van der Meer, | TU Delft, supervisor |
| | Dr. I. (Iuri) B.C.M. Rocha, | TU Delft, daily supervisor |
| | Dr. B. (Boyang) Chen, | TU Delft |

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft Delft University of Technology

# Abstract

Modern material systems with properly designed microstructures offer new avenues for engineering materials with advantageous mechanical properties and functionalities in various applications. With the increasing desire to design structures that have complex shapes or that are simply cheaper and more efficient, the demand for complex numerical simulations intensifies. Capturing the behaviour of composites relies on the interaction between the macroscopic and microscopic components, which can make direct numerical simulation too computationally expensive. Concurrent finite element analysis ($FE^2$) links the different length scales in a multiscale approach. In each integration point of the macroscale a representative volume element (RVE) is embedded to address the microscopic heterogeneity. However, the need to evaluate the micromodel many times will come with the drawback of high computational costs.

Various surrogate models have been proposed to accelerate computational models. One possible approach is employing machine learning techniques to substitute the evaluation of the RVE. Currently, artificial neural networks are being used in a wide variety of fields, due to their exceptional ability to recognize patterns. Although these techniques are able to construct models for complex input–output relations, their applications to mechanics of materials are still limited. These techniques are usually problem-dependent and may suffer from the danger of extrapolation beyond the original sampling space, e.g. different loading paths. This is not naturally resolved, mainly due to the loss of physics in the current machine learning models.

The newly proposed Deep Material Network (DMN) by Liu et al. [1] is a data-driven multiscale material modelling method that aims to resolve this issue by composing a network constructed from an assembly of mechanistic building blocks, where each building block resembles an analytical homogenization. Because the fitting parameters of this network are interpretable with physical meaning and because sampling the DMN still involves evaluation of classical constitutive relations, the loss of essential physics in minimized. The material network can effectively be trained using stochastic gradient descent with backpropagation, based on linear elastic data of an RVE.

The extrapolation capabilities to unknown loading paths are investigated with a matrix-inclusion composite, with the focus on the accuracy of the model for loading-unloading scenarios, and an extension to the training process is proposed where the fidelity of a trained DMN is assessed by generating and grading a yield stress envelope. The results show that a well trained DMN is capable of predicting the response for unseen loading paths involving unloading exceptionally well. The complete learning and extrapolation procedures of the DMN establish a reliable data-driven framework for concurrent multiscale material modeling and design.

# Contents


Page

List of Figures v

List of Algorithms vii

1 Introduction 1
- 1.1 Aim of the research and thesis outline . . . 2
- 1.2 Finite Element Method . . . 2
- 1.3 Concurrent multiscale analysis . . . 3
- 1.4 Neural networks . . . 4
- 1.5 Recurrent neural networks . . . 6
- 1.6 The new proposal . . . 8
  - 1.6.1 Creating intuition . . . 9

2 Deep Material Network 11
- 2.1 Mechanistic building block . . . 12
- 2.2 Network architecture . . . 12
- 2.3 Cost function . . . 13
- 2.4 Feedforward . . . 14
  - 2.4.1 Homogenization function $\mathfrak{h}_D$ . . . 14
  - 2.4.2 Rotation function $\mathfrak{r}_D$ . . . 15
- 2.5 Backpropagation . . . 15
  - 2.5.1 Gradients within a building block . . . 16
  - 2.5.2 Arrangement by layer . . . 17
- 2.6 Training . . . 19
- 2.7 Prediction and extrapolation . . . 19
  - 2.7.1 Building Block . . . 20
  - 2.7.2 Homogenization functions . . . 20
  - 2.7.3 Rotation functions . . . 21
  - 2.7.4 De-homogenization functions . . . 21
- 2.8 Computational implementation . . . 22
  - 2.8.1 New development . . . 22
  - 2.8.2 Generating training samples . . . 23
  - 2.8.3 Using the DMN online . . . 23

3 Analysis 25
- 3.1 Underlying mechanism . . . 25
  - 3.1.1 Replicate any topology . . . 25
  - 3.1.2 Validity for plastic extrapolation . . . 26
- 3.2 Parametric study . . . 27
  - 3.2.1 Depth of the network . . . 28
  - 3.2.2 Network regularization . . . 28
  - 3.2.3 Online step size . . . 29
- 3.3 Design of Experiments (DoE) . . . 29
  - 3.3.1 Orthotropic versus isotropic samples . . . 31
  - 3.3.2 Range of the training samples . . . 32
  - 3.3.3 Sample size . . . 33

4 Results 35
- 4.1 Elastic response of a realistic RVE . . . 35
- 4.2 Nonlinear plasticity with small strain . . . 37
  - 4.2.1 Yield stress envelope . . . 37

4.2.2 Extension to the training process . . . . . 39
4.2.3 Different loading paths . . . . . 40
4.2.4 Different hardening laws . . . . . 42
4.2.5 Computing time . . . . . 43

**5 Conclusions and future work** **45**
5.1 Conclusion . . . . . 45
5.2 Recommendations for future work . . . . . 46

# List of Figures


1.1 Multiscale modelling upscales information and allows for detailed physics representation [7]. . 2
1.2 The $FE^2$ approach links the macro- and microscale [8]. . . . . 3
1.3 Macro- and microscopic structures [12]. . . . . 3
1.4 Example of a neural network [18]. . . . . 4
1.5 Plots of polynomials having various orders $M$, shown as red curves (Bishop [20]). . . . . 5
1.6 Plots of the solutions obtained by minimizing the sum-of-squares error function using the $M = 9$ polynomial for $N = 15$ data points (left plot) and N = 100 data points (right plot) (Bishop [20]). . 5
1.7 Theoretical $\sigma$-$\varepsilon$ curve. . . . . 6
1.8 Dense/recurrent layer with 32 neurons, 100 thousand epochs. . . . . 6
1.9 Recurrent neural network [25]. . . . . 7
1.10 Eight stress-strain curves with different locations of unloading. . . . . 7
1.11 Prediction of the validation set after 100 thousand epochs. . . . . 8
1.12 Common hazards when dealing with neural networks [32]. . . . . 8
1.13 Schematic of the DMN-based concurrent multiscale structural analysis. Each integration point in the macroscale composite structure is linked to a microscale DMN model. The macroscale material responses (e.g., stress) are extracted from the top node of DMN, while the internal variables locate at the nodes in the bottom layer [34]. . . . . 9
1.14 Different breeds of dogs. . . . . 9
1.15 Graphical representation of the difference surrogate models. . . . . 10

2.1 Building Block. . . . . 12
2.2 Material Network. . . . . 12
2.3 Weight tree, visual explanation how the volume fractions are calculated. . . . . 13
2.4 In each building block there is a homogenization and a rotation. . . . . 14
2.5 Each node in bottom layer ($N = 2$) has its independent $\sigma$-$\varepsilon$ curve. . . . . 19
2.6 Each building block has a homogenization and a rotation function. . . . . 20
2.7 Via each building block the incremental strain is fed backwards. . . . . 21
2.8 Solution of an FE problem: Analysis flow and communication between modules and models. . 22

3.1 A DMN of depth $N = 1$ can represent a simple RVE exact. (Phase 1: $E = 5000, \nu = 0.2$, phase 2: $E = 5000, \nu = 0.3, \nu^{pl} = 0.39$.) . . . . 25
3.2 RVE and $\sigma$-$\varepsilon$ curve of situation A. . . . . 26
3.3 RVE and $\sigma$-$\varepsilon$ curve of situation B. . . . . 26
3.4 The summed up rotation of the information of each bottom node. Deactivated nodes are marked with a red cross and the size of the circle corresponds to its weight $z^j$. . . . . 27
3.5 The $\sigma - \varepsilon$ relation for the eight bottom nodes of a network of depth $N = 3$, corresponding to phase 1 (red) and phase 2 (black). . . . . 27
3.6 Mesh of a two-phased RVE resembling a fiber enforced material with fiber volume fraction of 27%. 28
3.7 Training (solid) and validation (dashed) error during training for different values for the regularization parameter $\lambda$. . . . . 29
3.8 Uniaxial tension test using a trained DMN ($N = 7$) with varying step sizes. . . . . 29
3.9 Uniaxial tension test to compare importance of the sampling variables. . . . . 31
3.10 Training error during training with only one sample (a) and online prediction (b). . . . . 31
3.11 Histories of average training errors and validation errors with a sample space limited to (a) isotropic matrices and (b) orthotropic elastic matrices. . . . . 32
3.12 Comparison between DMNs trained with an (a) isotropic and (b) orthotropic sample space. . . 32
3.13 Influence of the range in the ratio between the Young's moduli of the two materials when creating training data. . . . . 33

3.14 Training (solid) and validation (dashed) error during training with depth $N = 7$ for different limits to the ratio of the Young's moduli between the two materials in the training samples. . . . . 33
3.15 Training (solid) and validation (dashed) error with increasing sample size. . . . . 34

4.1 Mesh of a two-phased RVE resembling a fiber enforced material with fiber volume fraction of 50%. 35
4.2 Histograms of the error of the four batches. . . . . 36
4.3 Error distribution with on the x-axis the stiffness ratio; stiff fibers left, soft fibers right. . . . . 36
4.4 Input hardening curves for micromechanical plasticity model. [41] . . . . . 37
4.5 Yield stress envelopes. . . . . 37
4.6 Yield stress envelope with all four quadrants, for $\sqrt{\varepsilon_{xx}^2 + \varepsilon_{yy}^2} = 0.01$ & 0.04. . . . . 38
4.7 Yield stress envelope biaxial loading for six differently trained DMNs. . . . . 39
4.8 Six network of depth $N = 7$ are trained with the same $(z^j, \theta_i^j)$, but with training sets generated with a different random seed. . . . . 40
4.9 Uniaxial tension with single unloading. . . . . 40
4.10 Uniaxial tension with loading and unloading in stages. . . . . 41
4.11 Loading path combining tension and shear. . . . . 41
4.12 Two similar loading paths. . . . . 41
4.13 Two stress-strain graphs with related loading paths. . . . . 42
4.14 Uniaxial tension test. . . . . 42
4.15 Uniaxial tension with loading and unloading. . . . . 42
4.16 Single uniaxial loading with unloading, with J2-plasticity. . . . . 43
4.17 Uniaxial tension test. . . . . 43
4.18 Computational time versus number of fitting parameters. . . . . 44

# List of Algorithms

1 Updating the fitting parameters. . . . . 19
2 Implementation $FE^2$ . . . . . 23
3 Implementation DMN . . . . . 23
4 Procedure online phase during one loading step. . . . . 24

5 Extention of the training method. . . . . 39

# 1 Introduction

In the field of structural mechanics the complexity of the encountered problems increases, so there is an increasing demand for smart surrogate solving methods that can improve computational efficiency. A frequent cause for complexity is the need to be able to handle composite materials (e.g. fiber-reinfocred polymers), where the microscale interactions between the different materials dictate the behaviour of the overall structure. This makes solving an extensive finite element model cumbersome. A common method to differentiate and connect the multiple length scales is concurrent multiscale analysis, or $FE^2$ [2, 3]. In each integration point in the macro mesh an additional FE model is embedded, called a representative volume element (RVE), to embody the microscopic heterogeneities. This homogenization technique can still take excessive computation time, so surrogate models are used that replace the micromodel. One possibility to address this problem is with the use of machine learning techniques.

Machine learning is a subfield of artificial intelligence. Instead of relying on analytical derived instructions, it is a system through which computers use a massive set of data and apply algorithms to "train" the internal fitting parameters, relying on patterns found in the training data. Machine learning systems are able to quickly apply knowledge and train from large data sets to excel at facial recognition, speech recognition, object recognition, translation, and many other tasks. Machine learning touches industries spanning from government to education to health care. It can be used by businesses focused on marketing, social media, customer service, driverless cars, and many more. It is now widely regarded as a core tool for decision making. Artificial neural networks (ANNs) are one type of machine learning models, inspired by the structure and function of biological neural networks. They are a class of pattern matching methods that are commonly used for regression and classification problems but are really an enormous subfield comprised of hundreds of algorithms and variations for all manner of problem types.

For some applications artificial neural networks are the only way to find a relation between some input and what it ought to represent, e.g. handwriting recognition. ANNs can also be used for finding a relation which is analytically possible, but costs too much computing power to derive. The rise of machine learning, especially deep learning based on neural networks, has been continuously advancing the frontier of materials modelling and multiscale simulations. Discovering hidden physical interdependencies using data-driven models and boost material predictions had been around for a while (Bhadeshia [4], Ghaboussi et al. [5] used "the self-organizing capabilities of the neural network"). But when the progress in computing power advanced, so did the development of deep learning algorithms (e.g. Hochreiter and Schmidhuber [6]). Many of new studies are being done about how to use neural networks in modelling the behaviour of multiscale materials. Many types of neural networks were used, but a recent study by Liu et al. [1] incorporates analytical functions in the fundamental architecture of the neural network. Different from a generic neural network, all its fitting parameters are interpretable with physical meanings, enabling the extrapolation of learned model.

## 1.1. Aim of the research and thesis outline

The goal of this thesis is to investigate further the prospects of the new proposed DMN [1]. The following research questions and subquestions have been formulated to guide and assess the research.

1. Can a Deep Material Network be used to accurately translate microscopic behaviour in an RVE to the macroscopic level and can the DMN serve as a surrogate constitutive model for $FE^2$-based microstructural models to increase the computational efficiency?
2. How well are the extrapolating abilities to unknown loading paths, particularly unloading situations?
3. How does the performance of this model compare to other surrogate models?

**Subquestions**

1. What is the optimal setting for the (hyper) parameters of the DMN?
2. How case sensitive are these settings?
3. What accuracy can be expected in general?
4. Are there situations where the extrapolation capabilities come up short?

In this chapter background information is presented concerning multiscale modelling and surrogate modelling with the focus on machine learning variants. In chapter 2 the exact working of the new proposed deep material network will be described in detail. Chapter 4 starts with an analysis about the interpretation and the reliability of the results produced by the DMN. Thereafter, the network is used for a number of loading cases where the attention is aimed at the network's ability to treat loading-unloading behaviour.

## 1.2. Finite Element Method

With the Finite Element Method (FEM) the domain $\Omega$ is discretized into smaller elements that are connected with nodes with $N_{dof}$ degrees of freedom. With the use of partial differential equations (PDE) the laws of physics can be described. For non-trivial geometries analytical methods do not suffice, so with the discretization an approximation is made using numerical methods. The interpolation functions that correspond with the geometry and physics of the FE model are derived and the Neumann and Dirichlet boundary conditions are applied on the boundary $\Gamma$. The global equilibrium problem is solved by locating a displacement vector $\boldsymbol{u}$ that satisfies the system of equations. This is done by minimizing the residual force vector $\boldsymbol{r} \in \mathbb{R}^N$.

$$\boldsymbol{r} = \boldsymbol{f}^{\Omega} - \boldsymbol{f}^{\Gamma} = \boldsymbol{0} \tag{1.1}$$

The external and internal force vector, $\boldsymbol{f}^{\Gamma} \in \mathbb{R}^N$ and $\boldsymbol{f}^{\Omega} \in \mathbb{R}^N$ respectively, are determined by:

$$\boldsymbol{f}^{\Gamma} = \int_{\Gamma} \boldsymbol{N}^T \boldsymbol{t}^{\Gamma} d\Gamma, \qquad \boldsymbol{f}^{\Omega} = \int_{\Omega} \boldsymbol{B}^T \boldsymbol{\sigma}^{\Omega} d\Omega \tag{1.2}$$

with traction $\boldsymbol{t}^{\Gamma}$, body stress $\boldsymbol{\sigma}^{\Omega}$, the shape functions $\boldsymbol{N}$ and its spatial derivative $\boldsymbol{B}$. For nonlinear problems a direct solution is not attainable and an iterative method is used (e.g. Newton-Raphson method). The displacement vector $\boldsymbol{u}$ is iteratively corrected until Eq. (1.1) is approximated with sufficient accuracy:

$$\boldsymbol{u}_n = \boldsymbol{u}_{n-1} + \Delta \boldsymbol{u}_n = \boldsymbol{u}_{n-1} - \boldsymbol{K}_{n-1}^{-1} \boldsymbol{r}_{n-1} \tag{1.3}$$

where the global stiffness matrix $\boldsymbol{K}$ is derived using the material stiffness matrix $\boldsymbol{D}$:

$$\boldsymbol{K} = \frac{\partial \boldsymbol{f}^{\Omega}}{\partial \boldsymbol{u}} = \int_{\Omega} \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{B} d\Omega \tag{1.4}$$

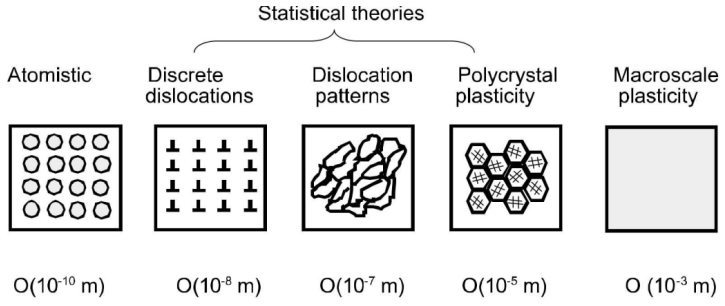



Figure 1.1: Multiscale modelling upscales information and allows for detailed physics representation [7].

## 1.3. Concurrent multiscale analysis

For most applications the material behaviour of a structure differs depending on how far is zoomed in. Metals are generally assumed to be isotropic, but when for instance cracks are analysed the heterogeneity at the lower length scales start to play a role, as seen in Figure 1.1. For polymers these characteristics arises from different mechanisms. Often it is meaningful to make a distinction between the macroscale and the microscale. For example when the structure is composed of multiple materials and their intrinsic boundaries need to be described, whilst keeping the constitutive models simple and tractable. Direct Numerical Simulation (DNS), where the entire domain is subdivided into a deeply dense mesh, is not manageable, so multiscale analysis ($FE^2$) is employed. Every integration point in the macroscopic mesh has a complete microscopic FE model embedded within. As illustrated in Figure 1.2 and Figure 1.3 the macro strain $\boldsymbol{\varepsilon}^{\Omega}$ in each integration point is transmitted to the micro mesh as boundary conditions. Thereafter, equilibrium is found with a solver and the resulting stiffness and stress are fed back to the macro level and are used as $\boldsymbol{D}^{\Omega}$ and $\boldsymbol{\sigma}^{\Omega}$ of that integration point.

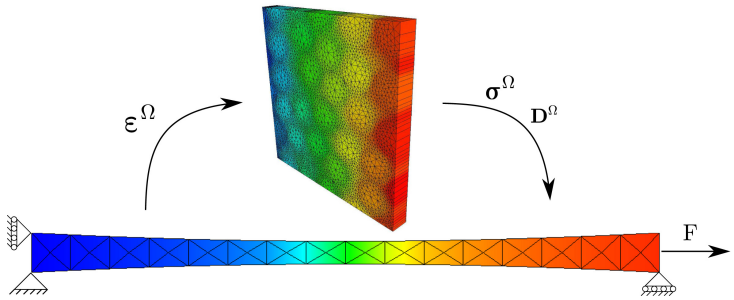



Figure 1.2: The $FE^2$ approach links the macro- and microscale [8].

The theory of multiscale analysis can be adopted for various problems. Guidault et al. [9] used the multiscale strategy for the computation of cracked structures. The main trends of computational homogenization are discussed in [10]. Kouznetsova studied the mechanical response of heterogeneous materials at large deformations and non-linear history dependent material behaviour [11]. She concluded that when using this micro-macro approach within the context of finite element implementation there is no need to specify the homogenized constitutive behaviour at the macroscopic integration points. Instead, this behaviour is determined through the detailed modelling of the microstructure.

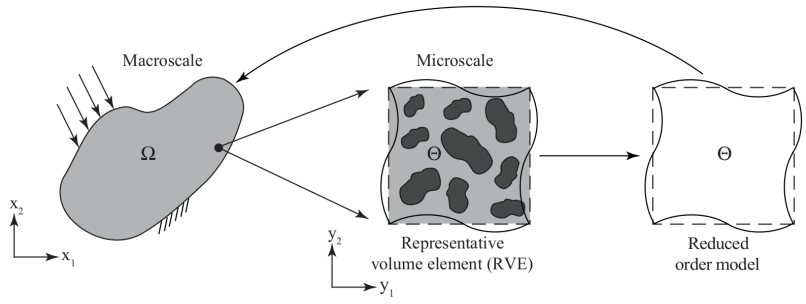



Figure 1.3: Macro- and microscopic structures [12].

The high computational cost of the micro models embedded at each integration point in the macroscale gave rise to the exploration of various surrogate models as a substitution for the constitutive relation of the RVE. With model order reduction (MOR) the number of degrees of freedom is reduced by by projecting the equilibrium equation onto a reduced space [13] by collecting snapshots and using dimensionality reduction techniques like proper generalized decomposition (PGD) [14] or proper orthogonal decomposition (POD). Another popular modelling approach consists of using micromechanical models to calibrate mesoscale constitutive models [15, 16, 17].

## 1.4. Neural networks

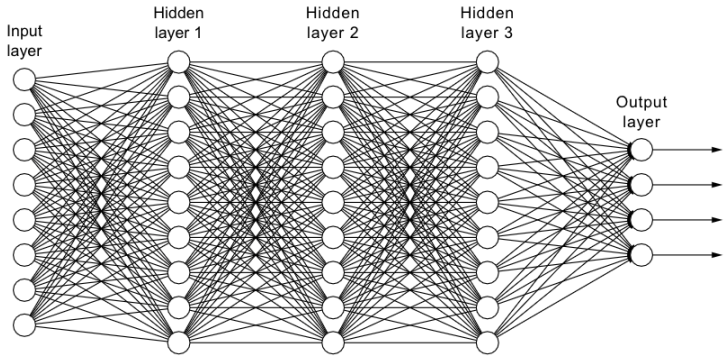



Figure 1.4: Example of a neural network [18].

Instead of finding a surrogate model that decreases the computational time by simplifying the micro models embedded at each integration point of the macroscale problem, it is also possible to only focus on the input and desired output and let a smart computer algorithm find a relation between the two. A forward neural network (FNN) in its simplest form has an input layer, one or more hidden layers and an output layer. Each neuron of a layer is connected to each neuron in the previous layer through a weight $w$. Together with a bias $b$ an activation $a$ in layer $l$ can be written as:

$$\boldsymbol{a}^l = \sigma\left(\sum \boldsymbol{W}^l \boldsymbol{a}^{l-1} + \boldsymbol{b}^l\right) \tag{1.5}$$

The choice of activation function, here denoted as $\sigma(\cdot)$, is case-dependent. For regression problems, the sigmoid function is a popular choice:

$$\sigma(v) = \frac{1}{1+e^{-v}} \tag{1.6}$$

The reason for its popularity is the fact that it scales the values between 0 and 1, and its derivative is easy to compute:

$$\sigma'(v) = \sigma(v) \cdot (1 - \sigma(v)) \tag{1.7}$$

The output of the network, the activations of the last layer $\boldsymbol{a}^L$, is a function of the input $\boldsymbol{x}$ values and all the weights and biases of the network. The weights and biases are independent fitting parameters which are to be trained:

$$\boldsymbol{a}^L = \boldsymbol{a}^L(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{b}) \tag{1.8}$$

In the case of a constitutive model the input layer receives the strain at a material point and the output layer generates the corresponding stress. It can be proven [19] that, given a dense layer with enough neurons, this structure can fit any continuous function within a given error tolerance.

To obtain the optimal values for the independent parameters the network is given a set of training data. The output of the network is compared to the desired output. To quantify how well the network is doing a cost function is used, e.g. based on the mean square error:

$$C(\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{b}) = \frac{1}{2n} \sum_n ||\boldsymbol{y}(\boldsymbol{x_n}) - \boldsymbol{a}^L(\boldsymbol{x_n}, \boldsymbol{W}, \boldsymbol{b})||^2 \tag{1.9}$$

where $n$ is the number of training samples. The goal is to minimize the cost function. The parameters $\boldsymbol{W}$ and $\boldsymbol{b}$ are initially given random values. After each training iteration the error is computed and backpropagated using stochastic gradient descent (SGD). The fitting parameters can be updated using the gradient of the cost function with respect to each fitting parameter and the learning rate $\eta$. If the network is trained enough such that the validation error is within the acceptable tolerance, it can be used in the online phase.

$$\boldsymbol{W}^{new} = \boldsymbol{W} - \eta \frac{\partial C}{\partial \boldsymbol{W}}, \qquad \boldsymbol{b}^{new} = \boldsymbol{b} - \eta \frac{\partial C}{\partial \boldsymbol{b}} \tag{1.10}$$

One significant drawback of neural networks is the danger of overfitting. This is where the fitting parameters are chosen such that the network does a good job fitting the training data, but will give bad results when extrapolating. Overfitting can have multiple causes. A network with many weights might be capable of capturing more complex behaviour, but only if not enough training points are used. In Figure 1.5 polynomials of different orders (in red) are fitted through the data points (in blue). A polynomial of order 9 fits the data points perfectly, but after a visual inspection it can be seen that the path of the polynomial in between the points is completely unlike the path the data points are sampled from.

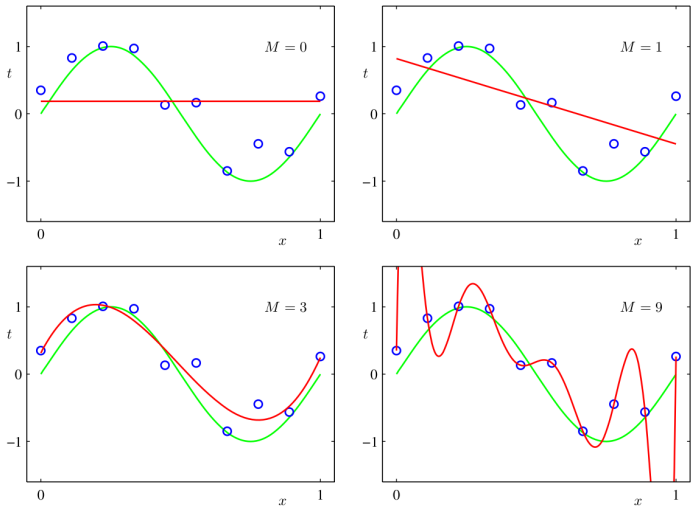



Figure 1.5: Plots of polynomials having various orders $M$, shown as red curves (Bishop [20]).

The number of data points has influence on the accuracy. There are techniques for handling limited data sets with neural networks [21], but usually enough data points are at hand. In Figure 1.6 we see that increasing the size of the data set reduces the overfitting problem.

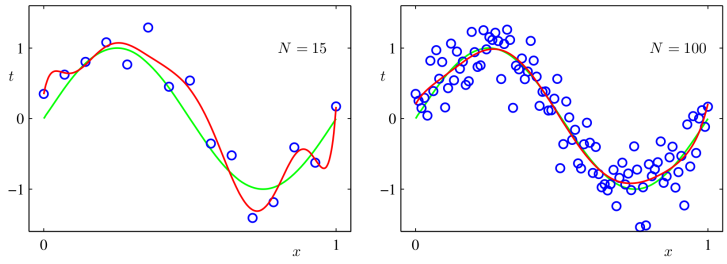



Figure 1.6: Plots of the solutions obtained by minimizing the sum-of-squares error function using the $M = 9$ polynomial for $N = 15$ data points (left plot) and N = 100 data points (right plot) (Bishop [20]).

Another to suppress overfitting is to adjust the cost function by adding a regularization term.

$$C(\boldsymbol{w},\boldsymbol{b})_{L_2} = C(\boldsymbol{w},\boldsymbol{b}) + \lambda \sum \boldsymbol{w}^2 \tag{1.11}$$

where $\lambda$ is a hyper-parameter. By applying this L2 regularization the squared sum of all the weights is added. One characteristic of overfitting is that some weights and biases become overly large. By including the weights in the cost function, the network is penalised for high weights. Another way to regularize the network is to add an extra dropout layer [22]. Each training iteration a randomly selected portion of the neurons in the dense layer is deactivated. Dropout has the effect of making the training process noisy, forcing nodes within a layer to probabilistically take on more or less responsibility for the inputs.

To investigate the abilities of the different types of neural networks a theoretical $\sigma$-$\varepsilon$ curve is considered. The material starts with elastoplastic behaviour, after which the it softens and eventually is loaded in the opposite

direction. One noteworthy feature is that there are no sharp transitions, which should will make it easier to find a good fit. The curve is made with:

$$\theta \in [0, \tfrac{3}{5}\pi], \qquad \varepsilon = \sin\theta^2, \qquad \sigma = 2\cos\theta\sin\theta \tag{1.12}$$

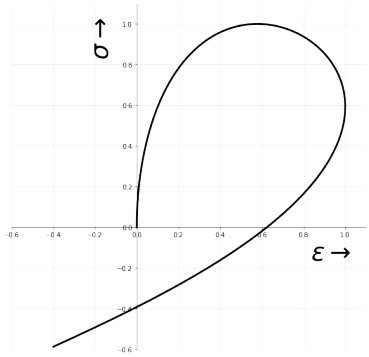
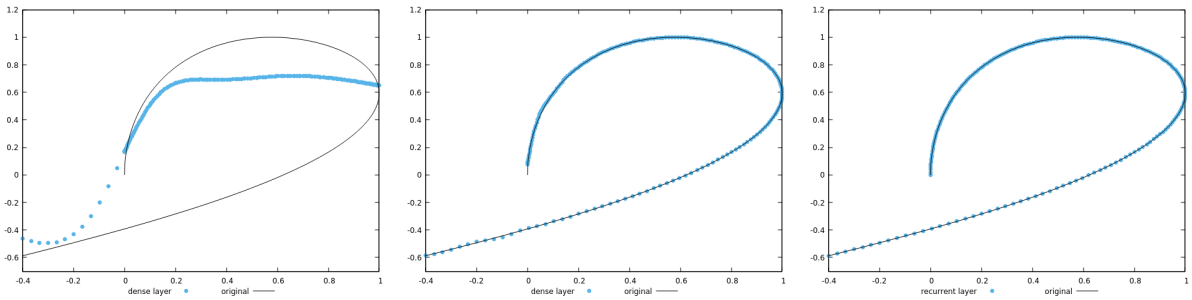



Figure 1.7: Theoretical $\sigma$-$\varepsilon$ curve.

First a feedforward neural network is used with a dense layer with 32 neurons, a sigmoid activation function and enough epochs, see Figure 1.8a. As expected the dense layer is unable to capture unloading, because it can not differentiate between multiple stress values that correspond to the same strain value. It is possible to capture unloading behaviour with a dense layer, but then more information must be given to the input layer. In Figure 1.8b the same network is used, but here the input layer has two neurons: the current strain and the strain of the previous data point, which will help distinguish an ascending from a descending path. Lastly a recurrent neural network is used in Figure 1.8c, with a similar result to the adjusted feedforward neural network. In this simple case an extra input parameter is sufficient for an FNN to work, but for more challenging situations that might not work.




(a) Input is current strain. (b) Input is current and previous strain. (c) Recurrent layer.

Figure 1.8: Dense/recurrent layer with 32 neurons, 100 thousand epochs.

More information about an intuitive understanding of the core concepts behind neural networks and deep learning can be found in Nielsen's interesting interactive online book [23] or the online textbook by Goodfellow et al. [24].

## 1.5. Recurrent neural networks

Where a feedforward neural network takes each input one at a time, a recurrent neural network (RNN) has the ability to process sequences of inputs. An RNN layer, as seen in Figure 1.9, computes its output based on the current input and the hidden state from the previous layer. This way it can use its memory to recognise patterns. Although it has more parameters to train, it's generally quicker and more robust than an FNN.

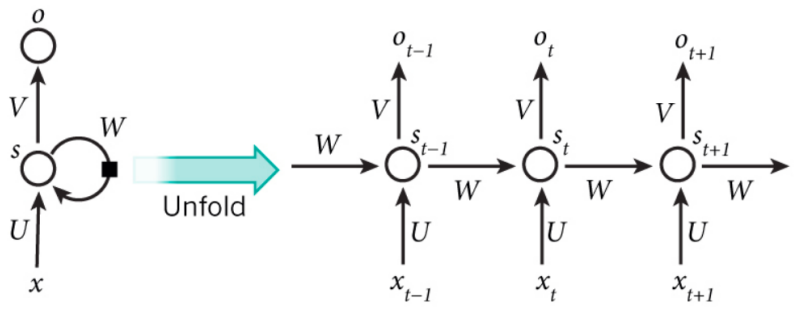


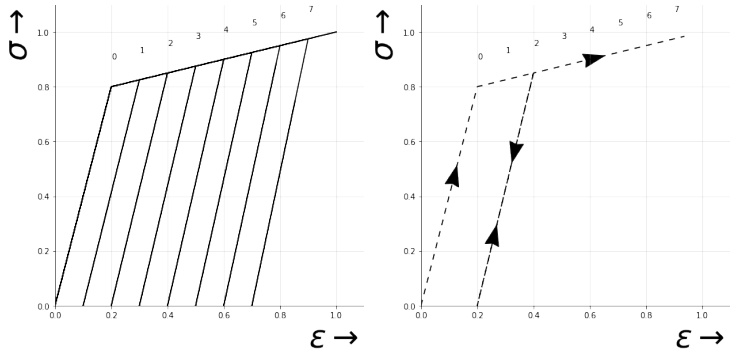

Figure 1.9: Recurrent neural network [25].

The hidden state $s_t$ is calculated based on the previous hidden state and the input at the current step. And $o_t$ is the output at step $t$. The functions $f$ and $g$ usually are a nonlinearity such as tanh, ReLU or softmax.

$$\begin{aligned} s_t &= f(Ux_t + Ws_{t-1}) \\ o_t &= g(Vs_t) \end{aligned} \tag{1.13}$$

Although recurrent neural networks have been proven to be a valuable contribution, there are drawbacks, like stated by Bengio et al. [26, 27]. A common obstacle with the standard RNN is the vanishing (or exploding) gradient problem. When training a vanilla RNN using backpropagation, the gradients which are backpropagated can tend to zero. Because of this, improved versions have been found like Long Short-Term Memory (LSTM) units, where certain crucial information can be kept for more time steps. Chung et al. [28] compared LSTM units and the more recently proposed Gated Recurrent Units (GRU) with RNNs on the tasks of polyphonic music modelling and speech signal modelling. These advanced recurrent units "are indeed better than more traditional recurrent units such as tanh units" [28]. Ghavamian and Simone [29] used a modified Long–Short Term Memory (mLSTM) unit to resemble a nonlinear finite element analysis to capture unloading and demonstrated its performance through a number of academic examples using strain-softening Perzyna viscoplasticity as the nonlinear material model at the micro level. Oishi and Yagawa [30] studied a method that utilizes deep learning to extract rules inherent in a computational mechanics application, which usually are implicit and sometimes too complicated to grasp from the large amount of available data.

As an example of the performance of an RNN an experiment is done with a fictitious material subject to a single loading-unloading tension test. Figure 1.10a shows eight stress-strain curves. The theoretical material is loaded up to yielding. During hardening the material is fully unloaded and reloaded. Each curve shows a different moment where the latter happens. Path $p$ follows the following road: origin, point 0, point $p$, down to $\sigma = 0$, point $p$ again and continues to the top right, like path 2 in Figure 1.10b. To test the ability of extrapolating, the training set consists of six of the eight paths and the remaining two paths are used for validating.

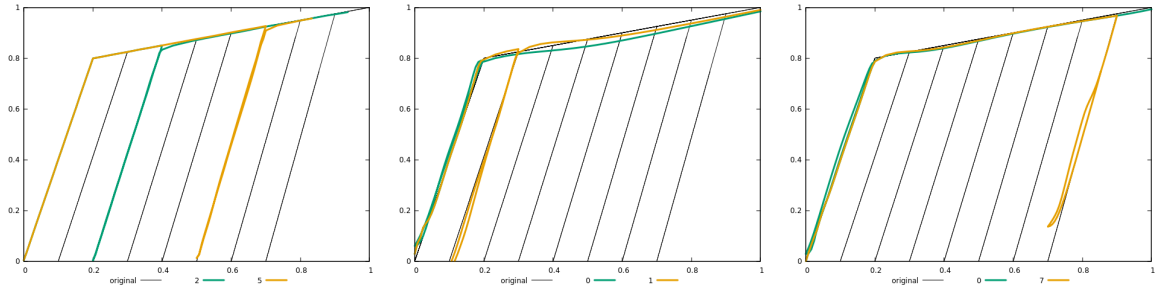



(a) All the eight paths. (b) Path 2.

Figure 1.10: Eight stress-strain curves with different locations of unloading.

If the validation set consists of paths that are within the boundaries of the training set, it is possible to reach a

validation error of less than 1%. If the paths that are chosen for validation are beyond the training domain, it can be impossible to reach a satisfactory validation error. In Figure 1.11 the result is shown after 100 thousand epochs with a recurrent layer of 128 units. It can be observed that the locations with the biggest error are the locations where there are discontinuities. This can be related to the fact that no knowledge about the material properties or about the loading procedure is given.

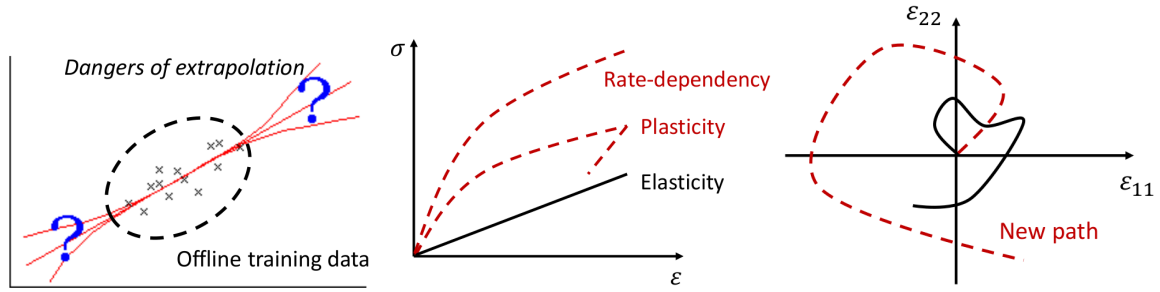


(a) Path 2 and 5. (b) Path 0 and 1. (c) Path 0 and 7.

Figure 1.11: Prediction of the validation set after 100 thousand epochs.

## 1.6. The new proposal

There is a diverse selection of surrogate models with different strengths and weaknesses. The problem with a neural network with only a dense layer is that it can only map continuous functions. It is not able to capture a graph that has multiple stress values for a single strain value, which is typical for unloading. A modified dense layer and a recurrent neural network are an improvement, but there is still the danger of extrapolation to unknown loading paths and material properties, as illustrated in Figure 1.12. The network will give unpredictable results when faced with cases outside the training scope. In addition, it is difficult to capture macroscopic behaviour when irregular complex morphologies, nonlinear history-dependent properties or large deformations are presented. Research is done to incorporate deep learning to solve path-dependent plasticity problems (e.g. Mozaffar et al. [31]), but there is still the loss of physics. With surrogate models based on model order reduction the microscopic interaction can be kept, but the improvements are often limited.




(a) Dangers of extrapolation. (b) Unknown material. (c) Unknown loading path.

Figure 1.12: Common hazards when dealing with neural networks [32].

In this thesis a new approach by Liu et al. [1] is investigated to obtain a simplified representation of a large-scale multiscale RVE with heterogeneous microstructures using a physics informed deep neural network, or Deep Material Network (DMN). The DMN is a collection of connected mechanistic building blocks with analytical homogenization solutions to describe complex overall material responses of an RVE. This avoids the loss of essential physics like in generic neural network. The network can be trained using linear elastic data collected from offline direct numerical simulations, after which this intelligent material model is capable of extrapolating the response of the RVE to unknown material and loading spaces. It is in principle valid for any material laws without the need for additional calibration or micromechanics assumption. This DMN with considerably fewer degrees of freedom than comparable neural networks has promising features and might open up new insights about physics informed deep learning.

The general working of this data-driven multiscale material modelling method is demonstrated for two dimensional RVE problems by Liu et al. [1]. Liu and Wu [33] extended their model for application to 3D heterogeneous materials with multiscale sources of nonlinearity such as particle-reinforced hyperelastic rubbers, polycrystalline materials, and elasto-plastic composites. In these papers the DMN is constructed to learn to predict the compliance matrix of an RVE for a certain two-scaled morphology, using as input the compliance matrices of the two phases. In this thesis a similar material network is built, but using stiffness matrices instead. In Figure 1.13 the basic concept is displayed and the next chapter goes into more detail about the exact working of the DMN.

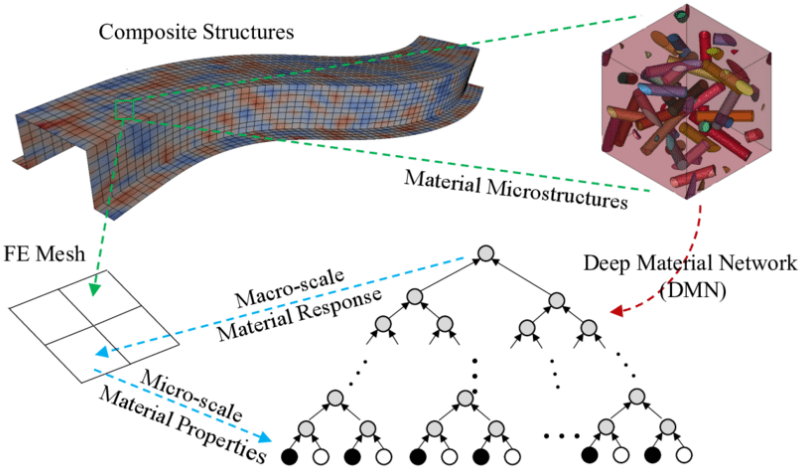



Figure 1.13: Schematic of the DMN-based concurrent multiscale structural analysis. Each integration point in the macroscale composite structure is linked to a microscale DMN model. The macroscale material responses (e.g., stress) are extracted from the top node of DMN, while the internal variables locate at the nodes in the bottom layer [34].

### 1.6.1. Creating intuition

To give the reader a rough understanding of how the new proposal acts in comparison to conventional neural networks, two cases are discussed, each from a different perspective.

**Dog versus Not-Dog** The DMN can philosophically be seen as a network trained with many pictures of standing dogs plus a biology lesson about anatomy. The result is that now the DMN can for example also recognize pictures of running dogs or pictures of dogs under a strange angle. But how far can you go? How distorted can the picture be to still be recognized as a dog? Can a neural network that has fitting parameters that are inherently linked to biological features, trained with a dataset consisting of only Tibetan terriers (Figure 1.14a) and accurately predict the dog-ness of pictures of a Pembroke Welsh Corgi (Figure 1.14b)?

(a) Tibetian terrier photos from Stanford dog datasheet [35].

(b) Pembroke Welsh Corgi.

Figure 1.14: Different breeds of dogs.

**Material models** In Figure 1.15 the constitutive relation is shown for three models. For $FE^2$ it is imaginable what is happening; the strain is imposed on the dense mesh within the integration point, where the stress is computed. The calculation of the stress in all the elements has a high computational cost, so a surrogate model is designed for the gap between the input ($\boldsymbol{\varepsilon}$) and the output ($\boldsymbol{\sigma}$) with the aid of machine learning. A standard neural network is sometimes seen as a black box. The designer is aware of the mathematics involved, but still the process can be difficult to follow. In the case of the new proposed DMN there is still a cryptic element to it, but due the fact that the training is done utilizing material laws, the black box becomes more transparent. Another major difference is the location where the box is placed. The DMN is not replacing the constitutive relation like a standard NN does, but it is only replacing the formulation of the stiffness matrix of the RVE.

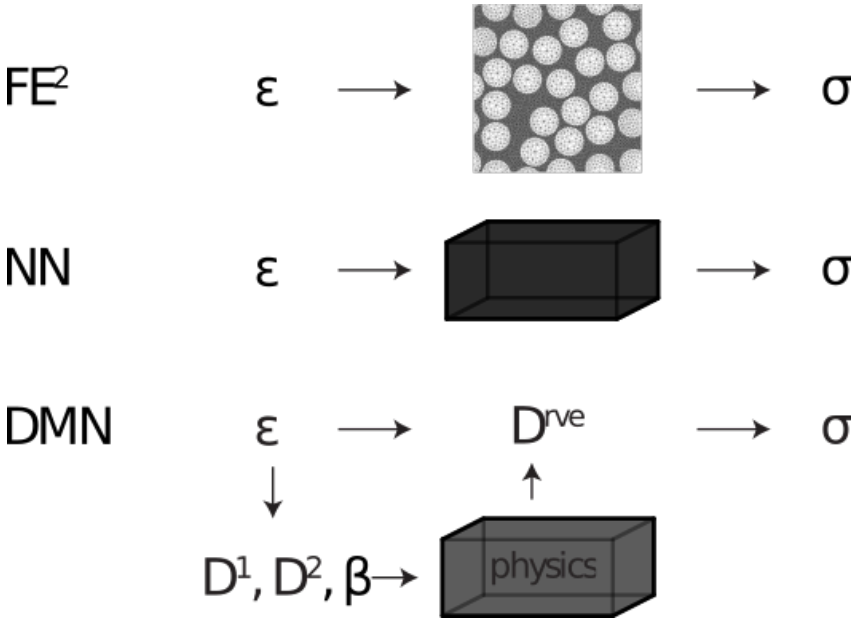



Figure 1.15: Graphical representation of the difference surrogate models.

# 2

# Deep Material Network

The DMN is used to predict the constitutive response of an RVE which contains elements of two different phases. After training for a certain topology of the RVE, the DMN is able to compute the stiffness of the RVE. So that means that where other neural networks are trained to predict the stress given the strain, this DMN is trained only to predict the stiffness matrix. During training it uses a dataset where each sample contains three elements; the stiffness matrix of the two phases and the stiffness matrix of the RVE. With a special tree-like structure subdivided into building blocks the network can be trained by optimizing the values of the fitting parameters; the activations $z^j$ and the rotations $\theta_i^j$.

The theory of the material network is developed for a two-phase linearly elastic building block in 2-dimensional plane strain condition. Under small-strain assumption, the stress and strain measures are the Cauchy stress $\boldsymbol{\sigma}$ and the infinitesimal strain $\boldsymbol{\varepsilon}$. They are related by the fourth-order stiffness tensor $\boldsymbol{D}$. The overall stress-strain relation of the building block can be expressed as

$$\boldsymbol{\sigma} = \boldsymbol{D} : \boldsymbol{\varepsilon} \tag{2.1}$$

For materials 1 and 2, we have

$$\boldsymbol{\sigma}^1 = \boldsymbol{D}^1 : \boldsymbol{\varepsilon}^1, \quad \boldsymbol{\sigma}^2 = \boldsymbol{D}^2 : \boldsymbol{\varepsilon}^2 \quad \text{and} \quad f_1 + f_2 = 1 \tag{2.2}$$

where $f_1$ and $f_2$ are the volume fractions of the two phases. To reduce the number of variables from now on the volume fraction of phase 1 will be denoted as $f$ and the volume fraction of the phase 2 as $1 - f$. For symmetric stress and strain tensors only three components are distinct. Using Mandel notation the stress and strain can be written as vectors.

$$\begin{aligned} \boldsymbol{\sigma} &= \left[\sigma_{11}, \sigma_{22}, \sqrt{2}\sigma_{12}\right]^T = \left[\sigma_1, \sigma_2, \sigma_3\right]^T \\ \boldsymbol{\varepsilon} &= \left[\varepsilon_{11}, \varepsilon_{22}, \sqrt{2}\varepsilon_{12}\right]^T = \left[\varepsilon_1, \varepsilon_2, \varepsilon_3\right]^T \end{aligned} \tag{2.3}$$

The factor of $\sqrt{2}$ ensures that the norm of the vector and the matrix remains the same. As a result the stiffness matrix can be written as a second order tensor.

$$\boldsymbol{\sigma} = \boldsymbol{D}\boldsymbol{\varepsilon}, \quad \boldsymbol{D} = \begin{bmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{bmatrix} \tag{2.4}$$

## 2.1. Mechanistic building block

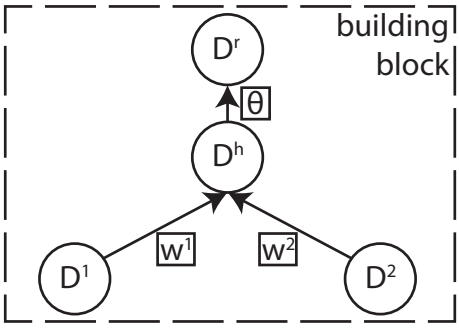



Figure 2.1: Building Block.

The main difference of a material neural network from a conventional neural network is that each parameter and each operation has a physical origin. This makes it possible to insert mechanical knowledge in the learning process. The network has a tree-like structure where each neuron is only dependent on two neurons in the layer below. This makes it tempting to not look at layers as the main subgroup, but see the network as an assembly of building blocks. Within such block two stiffness matrices are combined into one. As seen in Figure 2.1 the input of a building block are the stiffness matrices $\boldsymbol{D}^1$ and $\boldsymbol{D}^2$ of two neurons one layer down, their respective weights $w^1$ and $w^2$ and a rotation $\theta$. First $\boldsymbol{D}^1$ and $\boldsymbol{D}^2$ are combined using a homogenization operation $\mathfrak{h}_D$ to create $\boldsymbol{D}^h$. Then the rotation operation $\mathfrak{r}_D$ gives $\boldsymbol{D}^r$. The output will be a single stiffness matrix. When all the operations within one building block are defined, a network of any depth can be build.

## 2.2. Network architecture

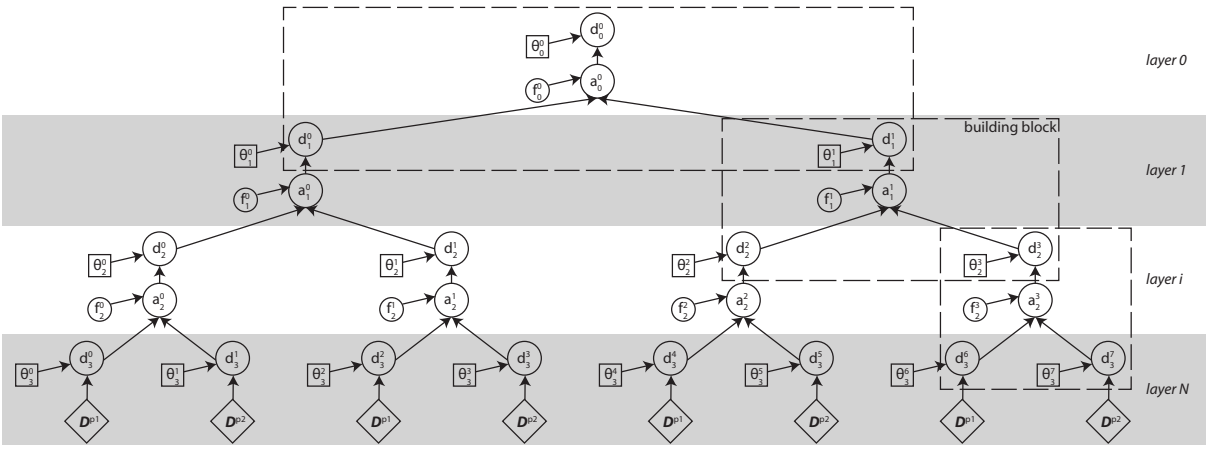



Figure 2.2: Material Network.

The depth of the network is denoted by $N$, the layers are indexed with $i$ and the neuron within each layer will be indexed with $j$ and $k$. The top layer (layer 0) is the output layer and the bottom layer is the input layer. The size of each layer $i$ is fixed at $2^i$ neurons. In Figure 2.2 a material network of depth $N = 3$ is shown. The rotated ($\boldsymbol{D}^r$) and homogenized ($\boldsymbol{D}^h$) stiffness matrices of neuron $j$ in layer $i$ are denoted as $d_i^j$ and $a_i^j$, respectively, and are second order tensors (3x3). The volume fraction $f_i^j$ and rotation $\theta_i^j$ are scalars. $\boldsymbol{D}^{p1}$ and $\boldsymbol{D}^{p2}$ are the stiffness matrices of the two phases in the RVE.

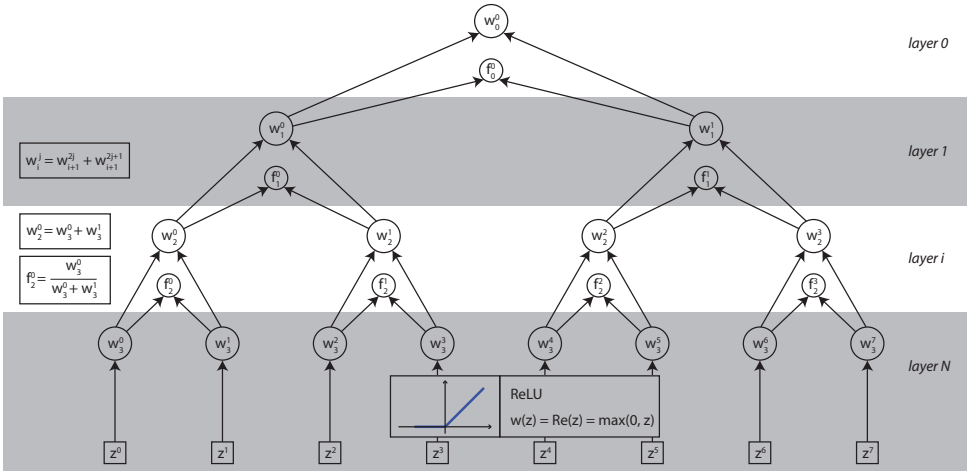



Figure 2.3: Weight tree, visual explanation how the volume fractions are calculated.

Along with the material network tree there is a weights tree (Figure 2.3) that gets updated every iteration as well and which determines the volume fractions. The weight of neuron $j$ in layer $i$ is the sum of the two weights one layer down and the volume fractions is their ratio. The weights can be any positive number while the volume fraction is a value between 0 and 1.

$$w_i^j = w_{i+1}^{2j} + w_{i+1}^{2j+1}, \qquad f_i^j = \frac{w_{i+1}^{2j}}{w_{i+1}^{2j} + w_{i+1}^{2j+1}} \tag{2.5}$$

Because of this additive relation in the weight tree, only the weights in the bottom layer $w_N^j$ are independent parameters. These are activated by an activation function and the activation parameter $z^j$. Here the rectified linear unit (ReLU) activation function is used.

$$w_N^j = Re(z^j) = \max(z^j, 0) \tag{2.6}$$

The fitting parameters of the total material network are the rotations $\theta_i^j$ in every neuron, and the activations $z^j$, which are only in the bottom layer. The total number of fitting parameters in a network of depth $N$ will be $3 \times 2^N - 1$.

For materials consisting of three or more phases the formation of the building block would go in a similar way, but the homogenization and the rotation operations would become more complex and more partial derivatives would be involved in the learning process. For a three-phase material the size of each layer would triple instead of double.

## 2.3. Cost function

The cost function that will be used here is based on the mean square error (MSE). The outcome of the RVE ($d_0^0$) is compared with the result of direct numerical simulations (DNS).

$$C_0(z,\theta) = \frac{1}{2} \frac{||\boldsymbol{D}^{dns} - \boldsymbol{D}^{rve}||^2}{||\boldsymbol{D}^{dns}||^2} \tag{2.7}$$

The cost function is normalized by dividing by $||\boldsymbol{D}^{dns}||^2$ to make it independent from the absolute values of the stiffness matrix. Because occasionally the activations $z^j$ have a tendency to grow without any significant benefit, a regularization term is added to the cost function. This will also decrease the effect of overfitting.

$$\begin{aligned} C(z,\theta,\lambda) &= C_0(z,\theta) + \lambda L(z) \\ L(z) &= \left( \sum_j Re(z^j) - \xi \right)^2 \end{aligned} \tag{2.8}$$

The hyper-parameters $\lambda$ and $\xi$ should be chosen carefully and their effect will be discussed later. In the next section the feedforward process is described. For training the network with gradient descent, the gradients of the cost function with respect to the fitting parameters are required. The backpropagation will be treated in paragraph 2.5. When all the mathematical tools are defined, the training process will be explained in paragraph 2.6. Finally the procedure is given how the trained network is used when applying it online.

## 2.4. Feedforward

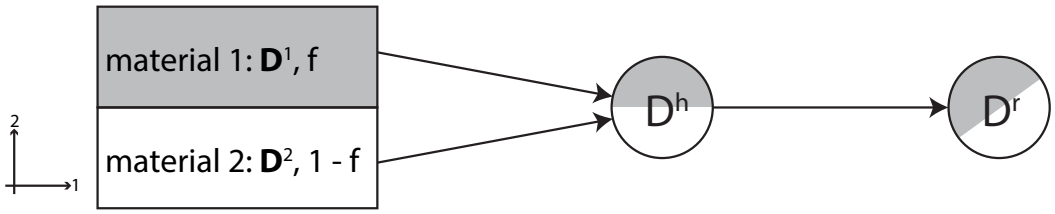



Figure 2.4: In each building block there is a homogenization and a rotation.

In one feedforward propagation the stiffness matrices in the bottom layer are propagated through the network to the output layer. The network is made up of building blocks, so here the process is explained for one building block which can then be extrapolated to the whole network. The structure of the building block consists of two operations as seen in Figure 2.4. For the same reason as for artificial neural networks, the operations should be simple to analyze. The first operation is homogenization given by the function $\mathfrak{h}_D$, which gives $\boldsymbol{D}^h$. The second operation is rotation given by $\mathfrak{r}_D$, which gives $\boldsymbol{D}^r$. This way it's possible to derive the gradients, which is essential during the network training process.

$$\boldsymbol{D}^h = \mathfrak{h}_D(\boldsymbol{D}^1, \boldsymbol{D}^2, f), \qquad \boldsymbol{D}^r = \mathfrak{r}_D(\boldsymbol{D}^r, \theta) \tag{2.9}$$

### 2.4.1. Homogenization function $\mathfrak{h}_D$

The analytical form of the homogenization function $\mathfrak{h}_D$ is derived based on the interfacial equilibrium conditions,

$$\sigma_2^1 = \sigma_2^2, \qquad \sigma_3^1 = \sigma_3^2 \tag{2.10}$$

and the kinematic constraint.

$$\varepsilon_1^1 = \varepsilon_1^2 \tag{2.11}$$

These three assumptions equate to vertical and shear stress being equal in both phases, as well as horizontal strain. Together with the following averaging equations for strain and stress there are nine conditions, enough to find a solution for the elements of the homogenized stiffness matrix.

$$\begin{aligned} \varepsilon_1 &= f\varepsilon_1^1 + (1-f)\varepsilon_1^2 \qquad \sigma_1 = f\sigma_1^1 + (1-f)\sigma_1^2 \\ \varepsilon_2 &= f\varepsilon_2^1 + (1-f)\varepsilon_2^2 \qquad \sigma_2 = f\sigma_2^1 + (1-f)\sigma_2^2 \\ \varepsilon_3 &= f\varepsilon_3^1 + (1-f)\varepsilon_3^2 \qquad \sigma_3 = f\sigma_3^1 + (1-f)\sigma_3^2 \end{aligned} \tag{2.12}$$

Eqs. (2.10) can be written in full

$$\begin{aligned} D_{21}^1\varepsilon_1^1 + D_{22}^1\varepsilon_2^1 + D_{23}^1\varepsilon_3^1 &= D_{21}^2\varepsilon_1^2 + D_{22}^2\varepsilon_2^2 + D_{23}^2\varepsilon_3^2 \\ D_{31}^1\varepsilon_1^1 + D_{32}^1\varepsilon_2^1 + D_{33}^1\varepsilon_3^1 &= D_{31}^2\varepsilon_1^2 + D_{32}^2\varepsilon_2^2 + D_{33}^2\varepsilon_3^2 \end{aligned} \tag{2.13}$$

Rewriting and substituting Eqs. (2.11) and (2.12) in Eq. (2.13) gives the following system of equations

$$\begin{bmatrix} \hat{D}_{22} & \hat{D}_{23} \\ \hat{D}_{32} & \hat{D}_{33} \end{bmatrix} \begin{bmatrix} \varepsilon_2^1 \\ \varepsilon_3^1 \end{bmatrix} = \begin{bmatrix} (1-f)\Delta D_{21}\varepsilon_1 + D_{22}^2\varepsilon_2 + D_{23}^2\varepsilon_3 \\ (1-f)\Delta D_{31}\varepsilon_1 + D_{32}^2\varepsilon_2 + D_{33}^2\varepsilon_3 \end{bmatrix}, \tag{2.14}$$

where

$$\hat{\boldsymbol{D}} = (1-f)\boldsymbol{D}^1 + f\boldsymbol{D}^2 \qquad \text{and} \qquad \Delta\boldsymbol{D} = \boldsymbol{D}^2 - \boldsymbol{D}^1 \tag{2.15}$$

The unknown strain components $\varepsilon_2^1$ and $\varepsilon_3^1$ of material 1 can now be expressed as a function of the strain components $\varepsilon_1$, $\varepsilon_2$, $\varepsilon_3$ of the overall strain.

$$\begin{bmatrix} \varepsilon_2^1 \\ \varepsilon_3^1 \end{bmatrix} = \left( \begin{bmatrix} \hat{D}_{22} & \hat{D}_{23} \\ \hat{D}_{32} & \hat{D}_{33} \end{bmatrix} \right)^{-1} \begin{bmatrix} (1-f)\Delta D_{21} & D_{22}^2 & D_{23}^2 \\ (1-f)\Delta D_{31} & D_{32}^2 & D_{33}^2 \end{bmatrix} \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix} \tag{2.16}$$

With the use of the definition of $\hat{\boldsymbol{D}}_{\boldsymbol{23}}$ and $\hat{\boldsymbol{s}}^{\boldsymbol{1}}$, Eq. (2.16) can be written as follows.

$$\hat{\boldsymbol{D}}_{\boldsymbol{23}} = \begin{bmatrix} \hat{D}_{22} & \hat{D}_{23} \\ \hat{D}_{32} & \hat{D}_{33} \end{bmatrix} \qquad \text{and} \qquad \hat{\boldsymbol{s}}^{\boldsymbol{1}} = \begin{bmatrix} (1-f)\Delta D_{21} & D_{22}^2 & D_{23}^2 \\ (1-f)\Delta D_{31} & D_{32}^2 & D_{33}^2 \end{bmatrix} \tag{2.17}$$

$$\begin{bmatrix} \varepsilon_2^1 \\ \varepsilon_3^1 \end{bmatrix} = \left(\hat{\boldsymbol{D}}_{\boldsymbol{23}}\right)^{-1} \hat{\boldsymbol{s}}^{\boldsymbol{1}}\, \boldsymbol{\varepsilon} \tag{2.18}$$

Eq. (2.18) gives the second and third row of the strain concentration tensor of material 1. The kinematic constraint yields the first row.

$$\boldsymbol{\varepsilon}^1 = \boldsymbol{s}^1\boldsymbol{\varepsilon}, \quad \text{with} \quad \boldsymbol{s}^1_{(1,:)} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \boldsymbol{s}^1_{([2,3],:)} = \left(\hat{\boldsymbol{D}}_{\boldsymbol{23}}\right)^{-1} \hat{\boldsymbol{s}}^{\boldsymbol{1}} \tag{2.19}$$

Rewrite the averaging equation for stress based on the definition of the second order tensor $\boldsymbol{s}^1$:

$$\boldsymbol{\sigma} = f\boldsymbol{\sigma}^1 + (1-f)\boldsymbol{\sigma}^2 = f\boldsymbol{D}^1\boldsymbol{\varepsilon}^1 + (1-f)\boldsymbol{D}^2\boldsymbol{\varepsilon}^2 = \boldsymbol{D}^2\boldsymbol{\varepsilon} - f\Delta\boldsymbol{D}\boldsymbol{\varepsilon}^1 = \left(\boldsymbol{D}^2 - f\Delta\boldsymbol{D}\boldsymbol{s}^1\right)\boldsymbol{\varepsilon} \tag{2.20}$$

The homogenized stiffness matrix before the rotation operation is obtained:

$$\boldsymbol{D}^h = \mathfrak{h}_D(\boldsymbol{D}^1, \boldsymbol{D}^2, f) = \boldsymbol{D}^2 - f\Delta\boldsymbol{D}\boldsymbol{s}^1 \tag{2.21}$$

### 2.4.2. Rotation function $\mathfrak{r}_D$

The final step in a building block is the rotation of the homogenized second-order stiffness tensor under Mandel notation. For a rotation $\theta$ the rotation matrix $\boldsymbol{R}$ is defined as

$$\boldsymbol{R}(\theta) = \begin{bmatrix} \cos^2\theta & \sin^2\theta & \sqrt{2}\sin\theta\cos\theta \\ \sin^2\theta & \cos^2\theta & -\sqrt{2}\sin\theta\cos\theta \\ -\sqrt{2}\sin\theta\cos\theta & \sqrt{2}\sin\theta\cos\theta & \cos^2\theta - \sin^2\theta \end{bmatrix}. \tag{2.22}$$

After rotation, the new stiffness matrix can be expressed as

$$\boldsymbol{D}^r = \mathfrak{r}_D(\boldsymbol{D}^h, \theta) = \boldsymbol{R}(-\theta)\boldsymbol{D}^h\boldsymbol{R}(\theta) \tag{2.23}$$

It is equivalent to deriving all the analytical homogenization and rotation functions based on the compliance tensor $\boldsymbol{C}$, which would even give neater expressions. The reason why we use the stiffness tensor $\boldsymbol{D}$ here is that usually the stiffness is known, whereas when using the compliance matrix it must first be inverted when calculating the stress.

## 2.5. Backpropagation

After the feedforward process the stiffness of the RVE that is computed by the network is compared with the correct stiffness. The error will be backpropagated through the network to calculate the derivative of the cost function with respect to every single fitting parameter, as seen in Eq. (2.24).

$$\nabla C(z,\theta,\lambda) = \left(\frac{\partial C}{\partial z}, \frac{\partial C}{\partial \theta}\right) = \left(\frac{\partial C_0}{\partial z} + \lambda\frac{\partial L}{\partial z}, \frac{\partial C_0}{\partial \theta}\right) \tag{2.24}$$

To formulate a derivative one has to find a path in Figure 2.2 from the top node ($d_0^0$) to the fitting parameter. This will result in a product of a chain of matrices. This will be more intricate that normal neural networks, because in a physics informed neural network the derivatives are derived using real constitutive relations. This results in a chain of fourth order matrices for every fitting parameter. In this section first the intermediate gradients will be presented, after which the total backpropagation process will be treated layer by layer.

### 2.5.1. Gradients within a building block

To compute the derivative of the strain concentration tensor $\boldsymbol{s}^1$ later, the derivative of the inverse of $\hat{\boldsymbol{D}}_{\mathbf{23}}$ with respect to any of the components is needed.

$$\frac{\partial \left(\hat{\boldsymbol{D}}_{\mathbf{23}}\right)^{-1}}{\partial(\cdot)} = -\left(\hat{\boldsymbol{D}}_{\mathbf{23}}\right)^{-1} \frac{\partial \hat{\boldsymbol{D}}_{\mathbf{23}}}{\partial(\cdot)} \left(\hat{\boldsymbol{D}}_{\mathbf{23}}\right)^{-1} \tag{2.25}$$

To find the derivative of $\boldsymbol{D}^r$ with respect to all the components of $\boldsymbol{D}^h$, the rotation operation is first rewritten in index notation.

$$D^r_{ij} = R_{ik}(-\theta)\, D^h_{kl}\, R_{lj}(\theta) \tag{2.26}$$

From this follows a fourth order tensor.

$$\frac{\partial D^r_{ij}}{\partial D^h_{kl}} = R_{ik}(-\theta) R_{lj}(\theta) \tag{2.27}$$

The derivative of $\boldsymbol{D}^r$ with respect to the rotation angle $\theta$ will be a second order tensor.

$$\begin{aligned} &\frac{\partial \boldsymbol{D}^r}{\partial \theta} = -\boldsymbol{R}'(-\theta)\boldsymbol{D}^h\boldsymbol{R}(\theta) + \boldsymbol{R}(-\theta)\boldsymbol{D}^h\boldsymbol{R}'(\theta) \\ &\text{with} \quad \boldsymbol{R}' = \begin{bmatrix} -\sin 2\theta & \sin 2\theta & \sqrt{2}\cos 2\theta \\ \sin 2\theta & -\sin 2\theta & -\sqrt{2}\cos 2\theta \\ -\sqrt{2}\cos 2\theta & \sqrt{2}\cos 2\theta & -2\sin 2\theta \end{bmatrix} \end{aligned} \tag{2.28}$$

The derivative of the homogenized stiffness matrix $\boldsymbol{D}^h$ with respect to the components of the *left* stiffness matrix one layer down, $\boldsymbol{D}^1$, will be a fourth order tensor.

$$\boldsymbol{D}^h = \boldsymbol{D}^2 - f\Delta\boldsymbol{D}\boldsymbol{s}^1, \qquad \frac{\partial \boldsymbol{D}^h}{\partial \boldsymbol{D}^1} = -f\left(\frac{\partial \Delta \boldsymbol{D}}{\partial \boldsymbol{D}^1} : \boldsymbol{s}^1 + \Delta\boldsymbol{D} : \frac{\partial \boldsymbol{s}^1}{\partial \boldsymbol{D}^1}\right) \tag{2.29}$$

The first row of $\boldsymbol{s}^1$ is $\begin{bmatrix}1 & 0 & 0\end{bmatrix}$, so any derivative will result in zeros in the first row. Using the definitions of Eq. (2.17), Eq. (2.25) and Eq. (2.19), the derivative of the second and third row of $\boldsymbol{s}^1$ will be:

$$\frac{\partial \boldsymbol{s}^1_{([2,3],:)}}{\partial \boldsymbol{D}^1} = \left(-\left(\hat{\boldsymbol{D}}_{\mathbf{23}}\right)^{-1} \frac{\partial \hat{\boldsymbol{D}}_{\mathbf{23}}}{\partial \boldsymbol{D}^1} \left(\hat{\boldsymbol{D}}_{\mathbf{23}}\right)^{-1}\right) : \hat{\boldsymbol{s}}^{\mathbf{1}} + \left(\hat{\boldsymbol{D}}_{\mathbf{23}}\right)^{-1} : \frac{\partial \hat{\boldsymbol{s}}^{\mathbf{1}}}{\partial \boldsymbol{D}^1} \tag{2.30}$$

The derivative of $\hat{\boldsymbol{D}}_{\mathbf{23}}$ with respect to the components of $\boldsymbol{D}^1$ are all zero, except for these

$$\frac{\partial \hat{\boldsymbol{D}}_{\mathbf{23}}}{\partial D^1_{22}} = \begin{bmatrix} 1-f & 0 \\ 0 & 0 \end{bmatrix} \quad \frac{\partial \hat{\boldsymbol{D}}_{\mathbf{23}}}{\partial D^1_{23}} = \begin{bmatrix} 0 & 1-f \\ 0 & 0 \end{bmatrix} \quad \frac{\partial \hat{\boldsymbol{D}}_{\mathbf{23}}}{\partial D^1_{32}} = \begin{bmatrix} 0 & 0 \\ 1-f & 0 \end{bmatrix} \quad \frac{\partial \hat{\boldsymbol{D}}_{\mathbf{23}}}{\partial D^1_{33}} = \begin{bmatrix} 0 & 0 \\ 0 & 1-f \end{bmatrix} \tag{2.31}$$

The derivatives of $\hat{\boldsymbol{s}}^{\mathbf{1}}$ with respect to the components of $\boldsymbol{D}^1$ are all zero, except or these

$$\frac{\partial \hat{\boldsymbol{s}}^{\mathbf{1}}}{\partial D^1_{21}} = \begin{bmatrix} -(1-f) & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \frac{\partial \hat{\boldsymbol{s}}^{\mathbf{1}}}{\partial D^1_{31}} = \begin{bmatrix} 0 & 0 & 0 \\ -(1-f) & 0 & 0 \end{bmatrix} \tag{2.32}$$

The derivative of the homogenized stiffness matrix $\boldsymbol{D}^h$ with respect to the components of the *right* stiffness matrix one layer down, $\boldsymbol{D}^2$, will also be a fourth order tensor and differs slightly from the previous derivation.

$$\boldsymbol{D}^h = \boldsymbol{D}^2 - f\Delta\boldsymbol{D}\boldsymbol{s}^1, \qquad \frac{\partial \boldsymbol{D}^h}{\partial \boldsymbol{D}^2} = \frac{\partial \boldsymbol{D}^2}{\partial \boldsymbol{D}^2} - f\left(\frac{\partial \Delta \boldsymbol{D}}{\partial \boldsymbol{D}^2} : \boldsymbol{s}^1 + \Delta\boldsymbol{D} : \frac{\partial \boldsymbol{s}^1}{\partial \boldsymbol{D}^2}\right) \tag{2.33}$$

The derivative of the the top row of $\boldsymbol{s}^1$ will be zeros again. Using the definitions of Eq. (2.17), Eq. (2.25) and Eq. (2.19), the derivative of the second and third row of $\boldsymbol{s^1}$ will be:

$$\frac{\partial \boldsymbol{s}^1_{([2,3],:)}}{\partial \boldsymbol{D}^2} = \left( -\left(\hat{\boldsymbol{D}}_{\boldsymbol{23}}\right)^{-1} \frac{\partial \hat{\boldsymbol{D}}_{\boldsymbol{23}}}{\partial \boldsymbol{D}^2} \left(\hat{\boldsymbol{D}}_{\boldsymbol{23}}\right)^{-1} \right) : \hat{\boldsymbol{s}}^{\mathbf{1}} + \left(\hat{\boldsymbol{D}}_{\boldsymbol{23}}\right)^{-1} : \frac{\partial \hat{\boldsymbol{s}}^{\mathbf{1}}}{\partial \boldsymbol{D}^2} \tag{2.34}$$

The derivative of $\hat{\boldsymbol{D}}_{\boldsymbol{23}}$ with respect to the components of $\boldsymbol{D}^2$ are all zero, except for these

$$\frac{\partial \hat{\boldsymbol{D}}_{\boldsymbol{23}}}{\partial D^2_{22}} = \begin{bmatrix} f & 0 \\ 0 & 0 \end{bmatrix} \qquad \frac{\partial \hat{\boldsymbol{D}}_{\boldsymbol{23}}}{\partial D^2_{23}} = \begin{bmatrix} 0 & f \\ 0 & 0 \end{bmatrix} \qquad \frac{\partial \hat{\boldsymbol{D}}_{\boldsymbol{23}}}{\partial D^2_{32}} = \begin{bmatrix} 0 & 0 \\ f & 0 \end{bmatrix} \qquad \frac{\partial \hat{\boldsymbol{D}}_{\boldsymbol{23}}}{\partial D^2_{33}} = \begin{bmatrix} 0 & 0 \\ 0 & f \end{bmatrix} \tag{2.35}$$

The derivatives of $\hat{\boldsymbol{s}}^{\mathbf{1}}$ with respect to the components of $\boldsymbol{D}^2$ are all zero, except or these

$$\begin{aligned} \frac{\partial \hat{\boldsymbol{s}}^{\mathbf{1}}}{\partial D^2_{21}} &= \begin{bmatrix} 1-f & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \frac{\partial \hat{\boldsymbol{s}}^{\mathbf{1}}}{\partial D^2_{22}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \frac{\partial \hat{\boldsymbol{s}}^{\mathbf{1}}}{\partial D^2_{23}} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \\ \frac{\partial \hat{\boldsymbol{s}}^{\mathbf{1}}}{\partial D^2_{31}} &= \begin{bmatrix} 0 & 0 & 0 \\ 1-f & 0 & 0 \end{bmatrix} \qquad \frac{\partial \hat{\boldsymbol{s}}^{\mathbf{1}}}{\partial D^2_{32}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad \frac{\partial \hat{\boldsymbol{s}}^{\mathbf{1}}}{\partial D^2_{33}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{2.36}$$

The derivative of the homogenized stiffness matrix $\boldsymbol{D}^h$ with respect to the volume fraction $f$ will be a second order tensor.

$$\boldsymbol{D}^h = \boldsymbol{D}^2 - f\Delta \boldsymbol{D}\boldsymbol{s}^1, \qquad \frac{\partial \boldsymbol{D}^h}{\partial f} = -\Delta \boldsymbol{D}\left( f\frac{\partial \boldsymbol{s}^1}{\partial f} + \boldsymbol{s}^1 \right) \tag{2.37}$$

The derivative of the the top row of $\boldsymbol{s}^1$ will be zeros again. Using the definitions of Eq. (2.17), Eq. (2.25) and Eq. (2.19), the derivative of the second and third row of $\boldsymbol{s^1}$ will be:

$$\frac{\partial \boldsymbol{s}^1_{([2,3],:)}}{\partial f} = \left(\hat{\boldsymbol{D}}_{\boldsymbol{23}}\right)^{-1} \begin{bmatrix} -\Delta D_{21} & 0 & 0 \\ -\Delta D_{31} & 0 & 0 \end{bmatrix} + \left( -\left(\hat{\boldsymbol{D}}_{\boldsymbol{23}}\right)^{-1} \Delta \boldsymbol{D}_{\boldsymbol{23}} \left(\hat{\boldsymbol{D}}_{\boldsymbol{23}}\right)^{-1} \right) \begin{bmatrix} (1-f)\Delta D_{21} & D^2_{22} & D^2_{23} \\ (1-f)\Delta D_{31} & D^2_{32} & D^2_{33} \end{bmatrix} \tag{2.38}$$

### 2.5.2. Arrangement by layer

As seen in Figure 2.2 each neuron consists of a rotated stiffness matrix, a homogenized stiffness matrix, a rotation and a volume fraction. The matrices are denoted as $d^j_i$ and $a^j_i$ and the derivative of the cost function with respect to them are the errors $\delta^j_i$ and $\alpha^j_i$, respectively, denoted as second order tensors.

$$\delta^j_i = \frac{\partial C_0}{\partial d^j_i}, \qquad \alpha^j_i = \frac{\partial C_0}{\partial a^j_i} \tag{2.39}$$

For the cost function in Eq. (2.7), the second order tensor $\delta_0$ in the output layer (layer 0) would look like

$$\delta^0_0 = \frac{\partial C_0}{\partial d^0_0} = \frac{d^0_0 - \boldsymbol{D}^{dns}}{||\boldsymbol{D}^{dns}||^2} \tag{2.40}$$

For $i \geq 1$, the error $\alpha^j_{i-1}$ and $\delta^k_i$ can be computed by

$$\begin{aligned} \alpha^j_{i-1} &= \frac{\partial C_0}{\partial d^j_{i-1}} : \frac{\partial d^j_{i-1}}{\partial a^j_{i-1}} = \delta^j_{i-1} : \frac{\partial d^j_{i-1}}{\partial a^j_{i-1}} \\ \delta^k_i &= \frac{\partial C_0}{\partial a^j_{i-1}} : \frac{\partial a^j_{i-1}}{\partial d^k_i} = \alpha^j_{i-1} : \frac{\partial a^j_{i-1}}{\partial d^k_i} \end{aligned} \tag{2.41}$$

where the fourth order tensor $\partial d_{i-1}^j / \partial a_{i-1}^j$ can be found in Eq. (2.27) and the fourth order tensor $\partial a_{i-1}^j / \partial d_i^k$ can be found in Eqs. (2.29) and (2.33). With the use of Eq. (2.28) the derivative of the cost function with respect to the rotations, which is a scalar, can now easily be computed.

$$\frac{\partial C_0}{\partial \theta_i^j} = \delta_i^j : \frac{\partial d_i^j}{\partial \theta_i^j} \tag{2.42}$$

The scalar-valued derivative of the cost function with respect to the activation parameters are more complicated, since every $a_i^j$ is dependent on $f_i^j$, which is dependent on two or more activation parameters. For example in Figure 2.3 $f_2^1$ is dependent on $z^2$ and $z^3$, while $f_1^1$ is dependent on $z^4$, $z^5$, $z^6$ and $z^7$. The derivative will be a summation of all the paths you can draw from $C_0$ to $z^j$.

$$\frac{\partial C_0}{\partial z^k} = \left( \sum_{i=0}^{(N-1)} \sum_{j=0}^{(2^i-1)} \alpha_i^j : \frac{\partial a_i^j}{\partial f_i^j} \times \frac{\partial f_i^j}{\partial w_N^k} \right) \times \frac{\partial w_N^k}{\partial z^k} \tag{2.43}$$

where $\partial a_i^j / \partial f_i^j$ can be found in Eq. (2.37). The derivative of the volume fraction of neuron $j$ in layer $i$ with respect to an arbitrary weight factor at the bottom layer, $w_N^k$:

$$\frac{\partial f_i^j}{\partial w_N^k} = \frac{1}{w_i^j} \left( \frac{\partial w_{i+1}^{2j}}{\partial w_N^k} - f_i^j \frac{\partial w_i^j}{\partial w_N^k} \right) \tag{2.44}$$

where the partial derivative of $w_i^j$ with respect to $w_N^k$ is

$$\frac{\partial w_i^j}{\partial w_N^k} = \begin{cases} 1, & \text{if} \quad j+1 = \left\lceil \frac{k+1}{2^{N-i}} \right\rceil \\ 0, & \text{otherwise} \end{cases} \tag{2.45}$$

The last term of Eq. (2.43) is the derivative of the activation function defined in Eq. (2.6).

$$\frac{\partial w_N^j}{\partial z^j} = \begin{cases} 1, & \text{if} \quad z^j > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.46}$$

In an attempt to give the reader a feeling of the essence of all the derivations up until now, the derivative of the cost function with respect to $\theta_0^0$, $\theta_2^1$ and $z^6$ are given.

$$\begin{aligned}
\frac{\partial C_0}{\partial \theta_0^0} &= \frac{\partial C_0}{\partial d_0^0} : \frac{\partial d_0^0}{\partial \theta_0^0} = \delta_0^0 : \frac{\partial d_0^0}{\partial \theta_0^0} \\
\frac{\partial C_0}{\partial \theta_2^1} &= \frac{\partial C_0}{\partial d_0^0} : \frac{\partial d_0^0}{\partial a_0^0} : \frac{\partial a_0^0}{\partial d_1^0} : \frac{\partial d_1^0}{\partial a_1^0} : \frac{\partial a_1^0}{\partial d_2^1} : \frac{\partial d_2^1}{\partial \theta_2^1} = \delta_2^1 : \frac{\partial d_2^1}{\partial \theta_2^1} \\
\frac{\partial C_0}{\partial z^6} &= \left( \alpha_0^0 : \frac{\partial a_0^0}{\partial f_0^0} \times \frac{\partial f_0^0}{\partial w_N^6} + \alpha_1^2 : \frac{\partial a_1^2}{\partial f_1^2} \times \frac{\partial f_1^2}{\partial w_N^6} + \alpha_2^3 : \frac{\partial a_2^3}{\partial f_2^3} \times \frac{\partial f_2^3}{\partial w_N^6} \right) \times Re'(z^6)
\end{aligned} \tag{2.47}$$

Finally, when applying the regularization introduced in Eq. (2.8), the gradients for the activation parameters are completed by adding the derivative of the regularization term, which will be equal for all $z^j$. In contrast to a standard neural network with regularization, the activation parameters all receive the same adjustment, while with an L2-regularization, which is commonly used in neural networks, the adjustment of one fitting parameters is not dependent on the value of the other fitting parameters.

$$\frac{\partial C}{\partial z^j} = \frac{\partial C_0}{\partial z^j} + \lambda \frac{\partial L}{\partial z^j} = \frac{\partial C_0}{\partial z^j} + \lambda \, 2 \left( \sum Re(z^j) - \xi \right) \tag{2.48}$$

## 2.6. Training

The training process itself is done in a similar manner as for standard neural networks. First the fitting parameters are initialized randomly:

$$\begin{aligned} z^{j(0)} &\in [0.2,\ 0.8] \\ \theta_i^{j(0)} &\in \left[-\tfrac{1}{2}\pi,\ \tfrac{1}{2}\pi\right] \end{aligned} \tag{2.49}$$

The algorithm then loops through the samples from the training data. For each sample the error is computed and backpropagated, so that the fitting parameters can be updated. When using stochastic gradient descent (SGD) the following equations would be used to update the fitting parameters.

$$\begin{aligned} \theta_i^{j(new)} &= \theta_i^j - \eta \frac{\partial C}{\partial \theta_i^j} \\ z^{j(new)} &= z^j - \eta \left( \frac{\partial C_0}{\partial z^j} + 2\lambda (\sum Re(z^j) - \xi) \right) \end{aligned} \tag{2.50}$$

A new hyper-parameter is introduced. The learning rate $\eta$ is case-dependent and should be chosen carefully, compromising training speed with stability. Here, the more advanced *Adam* algorithm (derived form Adaptive Moment Estimation) is used, proposed by Kingma and Ba [36], to expedite the training process. In this expansion of SGD, running averages of both the gradients and the second moments of the gradients are used. In Algorithm 1 the process of updating the fitting parameters once is summarized.

**Algorithm 1:** Updating the fitting parameters.

**Input:** $\boldsymbol{D}^1$, $\boldsymbol{D}^2$, $\boldsymbol{D}^{dns}$, $z^j$ and $\theta_i^j$.

1 **for** $i = N \rightarrow 0$ **do**

2 ⌊ **Feedforward:** Compute $a_i$ and $d_i$ using Eq. (2.21) and (2.23).

3 **Output error**: Compute $\delta_0$ using Eq. (2.40).

4 **for** $i = 1 \rightarrow N$ **do**

5 ⌊ **Backpropagate the error:** Compute $\alpha_{i-1}$ and $\delta_i$ using Eqs. (2.41).

6 **Gradients:** Compute the gradients of the cost function using Eqs. (2.42), (2.43) and (2.48).

**Output:** Update the fitting parameters $z^j$ and $\theta_i^j$ using SGD.

## 2.7. Prediction and extrapolation

When the DMN is trained and the optimum values for the fitting parameters $z$ and $\theta$ are found, the network can be used to predict unknown loading paths, even when the constitutive relation becomes nonlinear. Each node in the bottom layer acts as an individual material point. The degrees of freedom (DOFs) are the infinitesimal strain $\boldsymbol{\varepsilon}_N^j$ in each node in de bottom layer. Each node has its own loading path and internal variables.

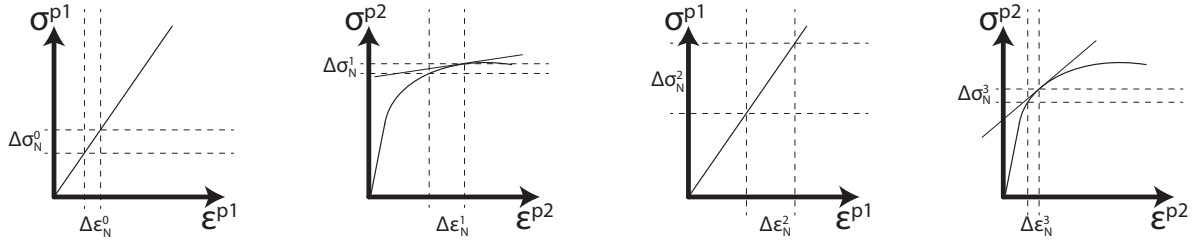



Figure 2.5: Each node in bottom layer ($N = 2$) has its independent $\sigma$-$\varepsilon$ curve.

Every load step on the macroscale an incremental strain $\Delta\boldsymbol{\varepsilon}^{rve}$ is given and an incremental stress $\Delta\boldsymbol{\sigma}^{rve}$ and a tangent stiffness $\boldsymbol{D}^{rve}$ is requested. To find the stress and stiffness of the RVE a configuration of the DOFs ($\boldsymbol{\varepsilon}_N^j$) is required that satisfies the macroscopic boundary conditions. The solution for this nonlinear RVE problem is solved iteratively. Each iteration consists of one forward propagation of the tangent stiffness and (if present) residual stress, and one backward propagation of the incremental strain.

Each iteration starts with adding the incremental strain in each bottom node to the strain it ended with during the previous loading step. (The first iteration starts with a incremental strain of zero, $\boldsymbol{\varepsilon}_N^{j(0)} = \mathbf{0}$.) The tangent stiffness, the incremental and residual stress can now be determined using the local constitutive law of each independent bottom node.

$$\begin{aligned}
\boldsymbol{D}_N^j &= \boldsymbol{D}_N^j(\boldsymbol{\varepsilon}_N^j, \Delta\boldsymbol{\varepsilon}_N^j, \boldsymbol{\sigma}_N^j, \boldsymbol{\beta}_N^j) \\
\Delta\boldsymbol{\sigma}_N^j &= \Delta\boldsymbol{\sigma}_N^j(\boldsymbol{\varepsilon}_N^j, \Delta\boldsymbol{\varepsilon}_N^j, \boldsymbol{\sigma}_N^j, \boldsymbol{\beta}_N^j) \\
\delta\boldsymbol{\sigma}_N^j &= \Delta\boldsymbol{\sigma}_N^j - \boldsymbol{D}_N^j \Delta\boldsymbol{\varepsilon}_N^j
\end{aligned} \tag{2.51}$$

The vector $\boldsymbol{\beta}_N^j$ can contain the internal variables if present. The tangent stiffness and the residual stress are propagated forward from the bottom layer ($i = N$) to the output layer ($i = 0$) according to the averaging equations in each building block. During this homogenization process the constitutive relation in every node of the material network is also determined. One can see this as laying out a path of breadcrumbs which can be followed back through the network, which makes the next step possible. In the second part of the iteration the incremental strain $\Delta\boldsymbol{\varepsilon}^{rve}$ is propagated back to the bottom layer using the constitutive relation at each node in the network, which results in a new distribution of the incremental strain $\Delta\boldsymbol{\varepsilon}_N^j$. Because of the rotations and volume fractions at each building block, the incremental strains in each node in the bottom layer will be different from each other. Some nodes may even already been deactivated during the training process. Convergence is reached when the relative difference of $\Delta\boldsymbol{\varepsilon}_N^j$ and $\Delta\boldsymbol{\varepsilon}_N^{j(prev)}$ is below a certain threshold. Upon convergence, the internal variables in each bottom node are updated so the next loading step can start.

### 2.7.1. Building Block

During the forward propagation of the tangent stiffness matrix $\boldsymbol{D}_N^j$ and the residual stress $\delta\boldsymbol{\sigma}_N^j$ the same concept is followed as during the training process. In Figure 2.6 the expanded building block is shown. The superscripts 1 and 2 stand for the child nodes if the building block, $h$ denotes the homogenized version and $r$ stands for the rotated version, which will be the output of the building block. For small-strain plasticity the homogenization and rotation operations for the tangent stiffness matrix stay the same as defined for the training process. The operations for the residual stress is derived below. Each building block consists of two operations; homogenization and rotation. Eq. (2.9) is now extended to

$$\begin{aligned}
&\boldsymbol{D}^h = \mathfrak{h}_D(\boldsymbol{D}^1, \boldsymbol{D}^2, f) \qquad \boldsymbol{D}^r = \mathfrak{r}_D(\boldsymbol{D}^h, \theta) \\
&\delta\boldsymbol{\sigma}^h = \mathfrak{h}_\sigma(\boldsymbol{D}^1, \boldsymbol{D}^2, \delta\boldsymbol{\sigma}^1, \delta\boldsymbol{\sigma}^2, f) \qquad \delta\boldsymbol{\sigma}^r = \mathfrak{r}_\sigma(\delta\boldsymbol{\sigma}^h, \theta)
\end{aligned} \tag{2.52}$$

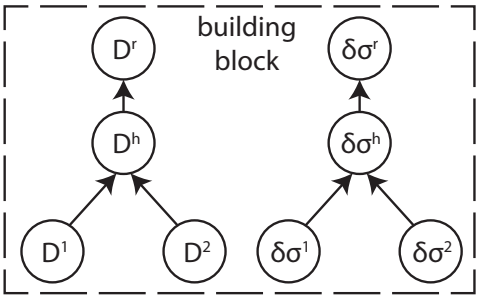



Figure 2.6: Each building block has a homogenization and a rotation function.

### 2.7.2. Homogenization functions

The analytical form of the homogenization function $\mathfrak{h}_D$ and $\mathfrak{h}_\sigma$ is derived based on the interfacial equilibrium conditions

$$\Delta\sigma_2^1 = \Delta\sigma_2^2, \quad \Delta\sigma_3^1 = \Delta\sigma_3^2 \tag{2.53}$$

and the kinematic constraints.

$$\Delta\varepsilon_1^1 = \Delta\varepsilon_1^2 \tag{2.54}$$

In the previous section $\mathfrak{h}_D$ is already defined as:

$$\boldsymbol{D}^h = \mathfrak{h}_D(\boldsymbol{D}^1, \boldsymbol{D}^2, f) = \boldsymbol{D}^2 - f\Delta\boldsymbol{D}\ \boldsymbol{s}^1 \tag{2.55}$$

To find $\mathfrak{h}_\sigma$ the average strain after homogenization $\Delta\boldsymbol{\varepsilon}^h$ is set to zero, so that the residual stress $\delta\boldsymbol{\sigma}^h$ is equal to the homogenized stress $\Delta\boldsymbol{\sigma}^h$.

$$\Delta\boldsymbol{\varepsilon}^h = \mathbf{0}, \qquad \Delta\boldsymbol{\sigma}^h = \delta\boldsymbol{\sigma}^h \tag{2.56}$$

This results in:

$$\begin{aligned} &\Delta\varepsilon_1^1 = \Delta\varepsilon_1^2 = 0 \\ &\Delta\varepsilon_2^2 = -\tfrac{f}{1-f}\Delta\varepsilon_2^1, \quad \Delta\varepsilon_3^2 = -\tfrac{f}{1-f}\Delta\varepsilon_3^1 \end{aligned} \tag{2.57}$$

The equilibrium conditions of Eq. (2.53) can be written in full.

$$\begin{aligned} &D_{22}^1\Delta\varepsilon_2^1 + D_{23}^1\Delta\varepsilon_3^1 + \delta\sigma_2^1 = D_{22}^2\Delta\varepsilon_2^2 + D_{23}^2\Delta\varepsilon_3^2 + \delta\sigma_2^2 \\ &D_{32}^1\Delta\varepsilon_2^1 + D_{33}^1\Delta\varepsilon_3^1 + \delta\sigma_3^1 = D_{32}^2\Delta\varepsilon_2^2 + D_{33}^2\Delta\varepsilon_3^2 + \delta\sigma_3^2 \end{aligned} \tag{2.58}$$

Substituting Eqs. (2.57) and rewriting gives:

$$\begin{bmatrix} (1-f)D_{22}^1 + fD_{22}^2 & (1-f)D_{23}^1 + fD_{23}^2 \\ (1-f)D_{32}^1 + fD_{32}^2 & (1-f)D_{33}^1 + fD_{33}^2 \end{bmatrix} \begin{bmatrix} \Delta\varepsilon_2^1 \\ \Delta\varepsilon_3^1 \end{bmatrix} = (1-f) \begin{bmatrix} \delta\sigma_2^2 - \delta\sigma_2^1 \\ \delta\sigma_3^2 - \delta\sigma_3^1 \end{bmatrix} \tag{2.59}$$

$$\begin{bmatrix} \Delta\varepsilon_2^1 \\ \Delta\varepsilon_3^1 \end{bmatrix} = (1-f)\left(\hat{\boldsymbol{D}}_{\mathbf{23}}\right)^{-1} \begin{bmatrix} \delta\sigma_2^2 - \delta\sigma_2^1 \\ \delta\sigma_3^2 - \delta\sigma_3^1 \end{bmatrix} \tag{2.60}$$

Rewrite the averaging equation for the residual stress:

$$\delta\boldsymbol{\sigma}^h = f\Delta\boldsymbol{\sigma}^1 + (1-f)\Delta\boldsymbol{\sigma}^2 = f\delta\boldsymbol{\sigma}^1 + (1-f)\delta\boldsymbol{\sigma}^2 + f\boldsymbol{D}^1\Delta\boldsymbol{\varepsilon}^1 + (1-f)\boldsymbol{D}^2\Delta\boldsymbol{\varepsilon}^2 \tag{2.61}$$

$$\delta\boldsymbol{\sigma}^h = \mathfrak{h}_\sigma(\delta\boldsymbol{\sigma}^1, \delta\boldsymbol{\sigma}^2, f) = f\delta\boldsymbol{\sigma}^1 + (1-f)\delta\boldsymbol{\sigma}^2 - f(1-f)\Delta\boldsymbol{D}_{(:,[2,3])}\left(\hat{\boldsymbol{D}}_{\mathbf{23}}\right)^{-1} \begin{bmatrix} \delta\sigma_2^2 - \delta\sigma_2^1 \\ \delta\sigma_3^2 - \delta\sigma_3^1 \end{bmatrix} \tag{2.62}$$

### 2.7.3. Rotation functions

After homogenization the tangent stiffness matrix and the residual stress are rotated.

$$\boldsymbol{D}^r = \mathfrak{r}_D(\boldsymbol{D}^h, \theta) = \boldsymbol{R}(-\theta)\boldsymbol{D}^h\boldsymbol{R}(\theta) \tag{2.63}$$

$$\delta\boldsymbol{\sigma}^r = \mathfrak{r}_\sigma(\delta\boldsymbol{\sigma}^h, \theta) = \boldsymbol{R}(-\theta)\delta\boldsymbol{\sigma}^h \tag{2.64}$$

### 2.7.4. De-homogenization functions

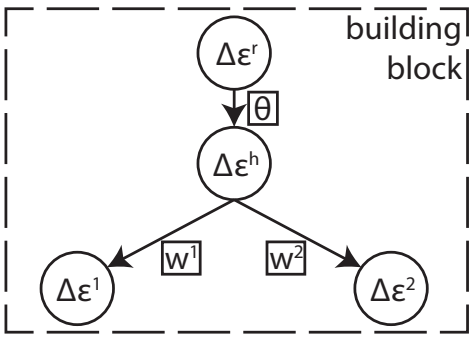



Figure 2.7: Via each building block the incremental strain is fed backwards.

In the de-homogenization process the macroscopic incremental strain $\Delta\boldsymbol{\varepsilon}^{rve}$ is backpropagated from the output layer to the bottom layer. Following again the arrangement of building blocks, the process is schematized in Figure 2.7. First the incremental strain is rotated back.

$$\Delta\boldsymbol{\varepsilon}^{h} = \boldsymbol{R}(\theta)\Delta\boldsymbol{\varepsilon}^{r} \tag{2.65}$$

With the use of the stiffness matrices in each node of the network, which are defined during the forward propagation, in each building block the incremental strain of the parent node can be split into the incremental strain of the two child nodes. When deriving the homogenization function $\mathfrak{h}_D$ in the section 2.4.1, the strain concentration tensor $\boldsymbol{s}^1$ is utilized to relate the strain of the left child node and the parent node, see Eq. (2.19). A similar relation can be found for the right child node.

$$\begin{aligned}\Delta\boldsymbol{\varepsilon}^{1} &= \boldsymbol{s}^{1}\Delta\boldsymbol{\varepsilon}^{h}\\ \Delta\boldsymbol{\varepsilon}^{2} &= \tfrac{1}{1-f}\Delta\boldsymbol{\varepsilon}^{h} - \tfrac{f}{1-f}\boldsymbol{s}^{1}\Delta\boldsymbol{\varepsilon}^{h}\end{aligned} \tag{2.66}$$

## 2.8. Computational implementation

The theory described in this chapter is implemented in a finite element code based on the *Jem-Jive* libraries, an open source *C++* programming toolkit [37]. In this paragraph, a short outline is presented how the implementation of the DMN is integrated in order to assist further development on the subject.

### 2.8.1. New development

The finite element code where the DMN is implemented is orderly compartmentalized, which enables easy permutation of fundamental components. Jive uses a division between *modules* that decide which tasks are done and *models* that decide how these tasks are done. All components interact via a global database, *globdat*, which stores all the runtime parameters. The three main components of each module are the *initialization*, *run* and *shutdown*. In a basic execution flow of a Jive application that solves an FE problem, Figure 2.8, the run function of each module is executed once for each load step.

One of the modules is a solver, which is in contact with a StressModel that performs the stress analysis. Finally, the StressModel requires a Material model, which receives the strain vector and returns the tangent stiffness matrix and the stress vector. The material model determines according to which material assumptions the constitutive relation in each element is calculated. To implement the DMN a new material model is build, called *DMNMaterial*, to compute the stress and stiffness. And the *DMNModel* is a class that implements the Deep Material Network. Both these classes are included on existing code available in the group.

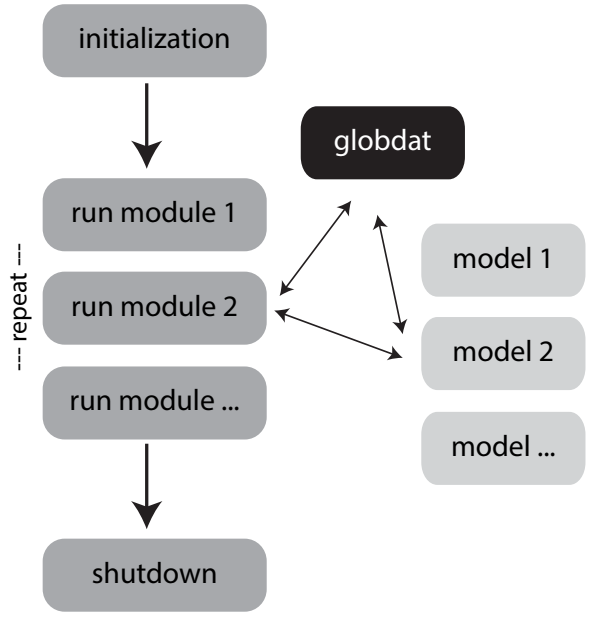



Figure 2.8: Solution of an FE problem: Analysis flow and communication between modules and models.

In the next chapters the accuracy of the DMN will be judged by comparing its results with the results generated with a full order model, computed with the $FE^2$ method. When using concurrent multiscale analysis, the StressModel uses a material model called *FE2*, which acts as an embedded micromodel. Inside, the elements of the RVE are divided into two (or more) groups, each of which in turn requires a StressModel and material model, etc. The macroscopic strain is imposed on the micromodel by utilizing periodic boundary conditions. The process is summarized in Algorithm 2. The source of the high computational cost originates in the high dimensionality of the components that are used in the constitutive relation in each macroscopic integration point. In Algorithm 3 the RVE can be considered as one element as the DMN has learned how to mimic the RVE response.

**Algorithm 2:** Implementation $FE^2$

```
for each load step do
    for each IP ∈ Ω do
        ε^Ω → BC^ω
        for each IP ∈ ω do
            ε^ω → σ^ω
        Find equilibrium in ω
        D^Ω, σ^Ω → IP^Ω
    Find equilibrium in Ω
```

**Algorithm 3:** Implementation DMN

```
for each load step do
    for each IP ∈ Ω do
        ε^Ω → DMNModel → D^Ω, σ^Ω
    Find equilibrium in Ω
```

### 2.8.2. Generating training samples

Training the network requires a large training dataset. In section 3.3 will be described how to properly tune the formation of the dataset, but one characteristic feature is that each sample is drawn from the linear elastic regime. A Python script was created that performs for each sample the following tasks:

1. Generate the material properties to create $\boldsymbol{D}^1$ and $\boldsymbol{D}^2$.
2. Update the properties file.
3. Execute the Jive application using one single load step with a full order $FE^2$ model with one element.
4. Retrieve the stiffness matrix of RVE, $\boldsymbol{D}^{rve}$.
5. Append the three stiffness matrices $\boldsymbol{D}^1$, $\boldsymbol{D}^2$ and $\boldsymbol{D}^{rve}$ to a file.

After that, samples are shuffled and divided into a training set and a validation set, so that it can be train as described in paragraph 2.6. Algorithm 1 shows the training procedure.

### 2.8.3. Using the DMN online

When the network is trained, it can be used as a material model just like any other; for each load step it receives a strain from the macroscopic integration point and returns the stiffness and stress. In Algorithm 4 the procedure of one loading step is summarized. For the evaluation of the constitutive relation in each bottom

node, the update function of the material models are called.

**Algorithm 4:** Procedure online phase during one loading step.

**Input:** Macroscopic strain of the RVE $\rightarrow \boldsymbol{\varepsilon}^{rve}$

**Present:** fitting parameters $(z,\theta)$, constitutive relations of phase 1&2 and their loading history

$\Delta\boldsymbol{\varepsilon}^{rve} = \boldsymbol{\varepsilon}^{rve} - \boldsymbol{\varepsilon}^{rve(prev)}$ (this one stays fixed)

$\boldsymbol{\varepsilon}_N^j = \boldsymbol{\varepsilon}_N^{j(prev)}$

$\Delta\boldsymbol{\varepsilon}_N^j = 0$, $\Delta\boldsymbol{\varepsilon}_{N0}^j = 0$

**for** *until converged* **do**

- $\boldsymbol{\varepsilon}_N^j, \Delta\boldsymbol{\varepsilon}_{N0}^j \rightarrow \boldsymbol{D}_N^j, \boldsymbol{\sigma}_N^j, \delta\boldsymbol{\sigma}_N^j$ (evaluate constitutive relation in bottom layer)
- $\boldsymbol{D}_N^j \rightarrow \boldsymbol{D}_i^j,\ \boldsymbol{D}^{rve}$ (propagate tangent stiffness)
- $\delta\boldsymbol{\sigma}_N^j \rightarrow \delta\boldsymbol{\sigma}_i^j,\ \delta\boldsymbol{\sigma}^{rve}$ (propagate residual stress)
- Evaluate macroscopic BCs
- $\Delta\boldsymbol{\varepsilon}^{rve},\ \boldsymbol{D}_i^j \rightarrow \Delta\boldsymbol{\varepsilon}_N^j$ (backpropagate incremental strain)
- Check if $|\Delta\boldsymbol{\varepsilon}_N^j - \Delta\boldsymbol{\varepsilon}_{N0}^j| <$ value (check if converged)
- $\Delta\boldsymbol{\varepsilon}_{N0}^j = \Delta\boldsymbol{\varepsilon}_N^j$

$\Delta\boldsymbol{\sigma}^{rve} = \boldsymbol{D}^{rve}\Delta\boldsymbol{\varepsilon}^{rve} + \delta\boldsymbol{\sigma}^{rve}$

$\boldsymbol{\sigma}^{rve} = \boldsymbol{\sigma}^{rve,(prev)} - \Delta\boldsymbol{\sigma}^{rve}$

**Output:** $\boldsymbol{\sigma}^{rve}, \boldsymbol{D}^{rve}$

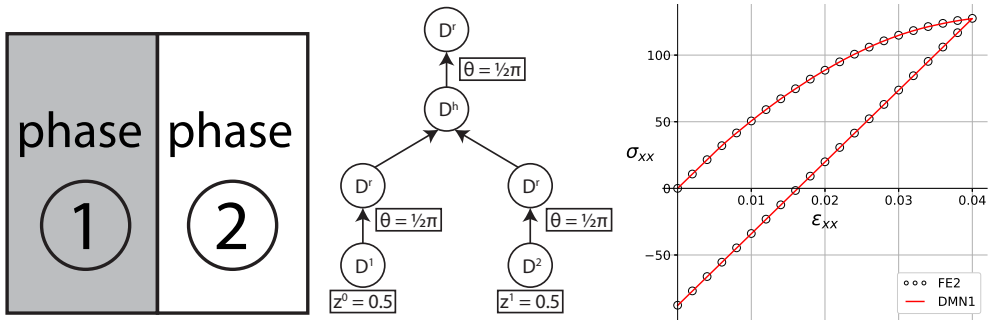
# 3

# Analysis

Now that the structure and the theoretical derivation of the Deep Material Network is defined, the underlying workings will be addressed in more detail to get a better understanding of the mechanism that drives the network. Before applying the network on a case, it is important to pay attention to the values of all the (hyper) parameters. There is a multitude of choices to be made when designing a robust and reliable network. Some of them purely relate to the efficiency of the training algorithms, while others are linked to the fidelity of the network. In the next section a deeper understanding of the inner working of the network will be investigated. After that, the influence of some of the parameters will be discussed and in section 3.3 the training and validation datasets are analyzed.

## 3.1. Underlying mechanism

### 3.1.1. Replicate any topology



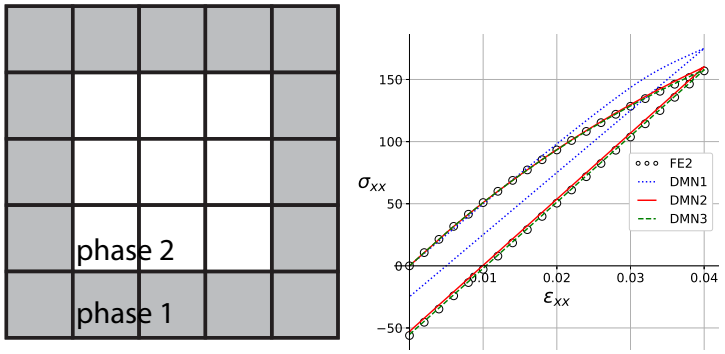

(a) Simple RVE. (b) DMN of depth $N = 1$. (c) Uniaxial loading and unloading.

Figure 3.1: A DMN of depth $N = 1$ can represent a simple RVE exact. (Phase 1: $E = 5000, \nu = 0.2$, phase 2: $E = 5000, \nu = 0.3, \nu^{pl} = 0.39$.)

In each two-layer building block an analytical solution is derived for the rudimentary combination like depicted in Figure 2.4. An exact solution is employed for homogenization (using a volume fraction $f$) and rotation (using an angle $\theta$). For an RVE with a different, more complicated topology an accumulation of building blocks is needed to represent its behaviour. With building blocks that are based on a rectangle that is partitioned horizontally, it is possible to find an exact representation of an RVE that is a square that is split vertically. In fact, a DMN of depth $N = 1$ is sufficient, as seen in Figure 3.1. (Due to the method of how the tree-like structure is assembled, the bottom layer gets an extra set of rotations, hence the three $\theta$'s.) Figure 3.1c shows that the result of the DMN of depth $N = 1$ is equal to that of a direct numerical solution for a uniaxial loading and unloading test. The cyclic nature of the the rotations $\theta$ ($\frac{1}{2}\pi \sim \frac{3}{2}\pi$) indicates that there are multiple solutions and when a deeper network is trained for this simple configuration, even more possible outcomes can give an exact rendition. One possible formation of the fitting parameters $(z^j, \theta_i^j)$ is shown in Figure 3.1b. $\boldsymbol{D}^1$ and $\boldsymbol{D}^2$ are each rotated 90°, combined by the (exact) homogenization operation and rotated back. For a block where

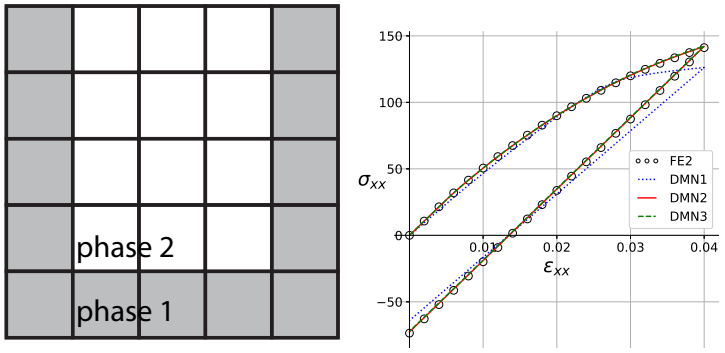
the vertical split is off-center, corresponding values for $z^0$ and $z^1$ should be used. It is worth emphasizing that not the absolute value of these activation parameters is of importance, but their ratio.

The fact that for a two-element block an exact solution can be found for both a horizontal split and a vertical split would argue that theoretically any RVE could be represented with a deep enough network, therefore using small enough two-element blocks. What the ideal depth for an RVE is beyond which no significant improvement is distinguished, is not necessarily clear. Take the following two (rather simple) RVEs that are tested with ascending depths with a uniaxial loading-unloading test. The one in Figure 3.2 has an optimal depth of $N = 3$, while a DMN of depth $N = 2$ is sufficient to capture the behaviour of the RVE in Figure 3.3. Possibilities have been searched intensively where a trained DMN would not be able to capture the response of the RVE, for any depth, but without result. Another way where the DMN could have a flaw is if there exists a pair of RVEs that behave identical in the elastic regime, but behave differently when plasticity comes into effect. But as it stands, the DMN seems to be indeed capable of representing general microscopic topologies, although we can not provide rigorous proof at this point.




(a) RVE. (b) Uniaxial loading and unloading.

Figure 3.2: RVE and $\sigma$-$\varepsilon$ curve of situation A.




(a) RVE. (b) Uniaxial loading and unloading.

Figure 3.3: RVE and $\sigma$-$\varepsilon$ curve of situation B.

### 3.1.2. Validity for plastic extrapolation

To obtain the tangent stiffness and the residual stress belonging to the macroscopic integration point at each load step, an iterative process starts where the incremental strain is propagated to the bottom layer ($i = N$), the constitutive relations of each bottom node are evaluated and its output is fed back to the top layer ($i = 0$). Just like every element in an FE model is at a different strain level, each bottom node of a DMN also behaves independently of each other, as stated in section 2.7. The information that is propagated up and down gets rotated at each layer, but interestingly, if all rotations to each active bottom node are summed up, the result is always close to a multiple of $\pi$ (assuming that the network is well trained). In Figure 3.4 the total rotation for each bottom node corresponding to phase 1 and 2 is plotted for a DMN of three different depths, trained on

the RVE shown in Figure 3.3. A red cross indicates a deactivated node and the size of the black circles coincide with the value of the activation parameter $z^j$ of the corresponding node.

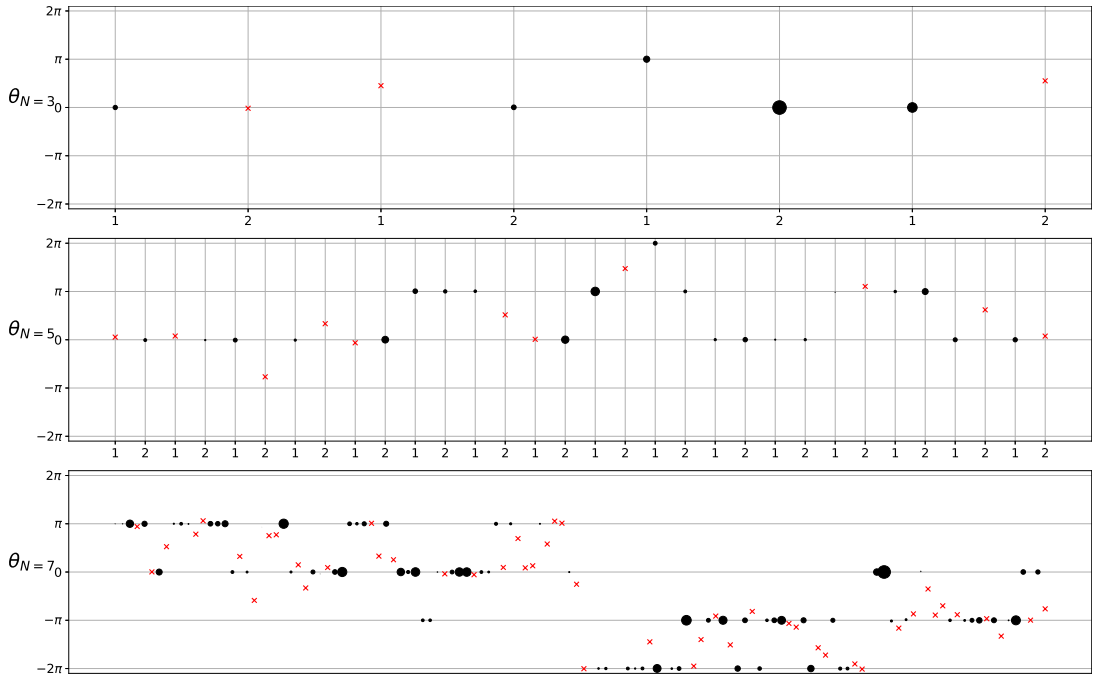



Figure 3.4: The summed up rotation of the information of each bottom node. Deactivated nodes are marked with a red cross and the size of the circle corresponds to its weight $z^j$.

The fact that each node is aligned in the same orientation allows us to easily analyse their individual stress paths. Inspection of the stress-strain curves related the the active bottom nodes of a DMN of depth $N = 3$, presented in Figure 3.5, show that each node indeed follows its own course. For a DMN trained for a more intricate RVE, like the ones used later in this thesis, the dissimilarity becomes even more evident. Besides the notion that the fitting parameters $(z^j, \theta_i^j)$ have physical meaning, this is where the physics is embedded and where the Deep Material Network is fundamentally different from conventional neural networks. There are still real material models being solved, with their individual material properties and history parameters. The neural network only acts as a translation from the multiple simple material nodes to the full scale model.

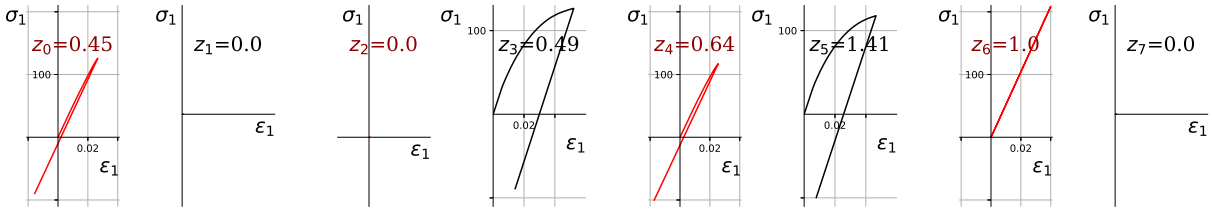



Figure 3.5: The $\sigma - \varepsilon$ relation for the eight bottom nodes of a network of depth $N = 3$, corresponding to phase 1 (red) and phase 2 (black).

## 3.2. Parametric study

In this section the influence of the network hyper parameters is investigated on a fixed example. The mesh of a two-phased RVE as shown in Figure 3.6 is used to train a network for 1000 epochs using minibatches of 32 samples, with a depth of $N = 7$ and regularization factors $\lambda = 0.001$ and $\xi = 2^{N-1}$. For training a dataset of 1000 samples is used of which 20% is dedicated for validation, with a sample space described in Section 3.3. When using the trained network online the RVE is tested under uniaxial tension up to $\varepsilon_{11} = 0.01$, where the fibers (phase 1) are chosen to remain linear elastic and the matrix (phase 2) follows a bilinear hardening law, seen in Eq. (3.1). For the matrix the pressure-dependent elastoplastic model proposed by Melro et al. [38, 39] is used.

For the material properties we start with two hypothetical phases ($p1$ and $p2$) with $E^{p1} = 500, \nu^{p1} = 0.19, E^{p2} = 100, \nu^{p2} = 0.3$.

$$\sigma_Y = \begin{cases} 0.1 + 5\varepsilon_{pl}, & \varepsilon_{pl} < 0.008 \\ 0.124 + 2\varepsilon_{pl}, & \varepsilon_{pl} \geq 0.008 \end{cases} \tag{3.1}$$

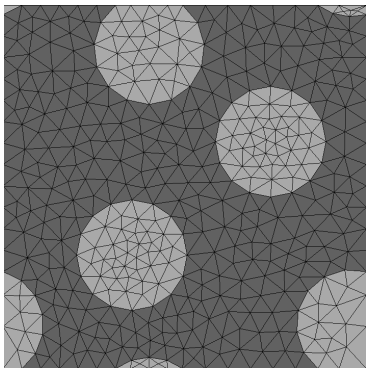


Figure 3.6: Mesh of a two-phased RVE resembling a fiber enforced material with fiber volume fraction of 27%.

### 3.2.1. Depth of the network

The choice for the depth $N$ of the network highly depends on the RVE the DMN is trained for. In general a deeper network is better capable of predicting the response, but there exists a point of diminishing gains after which increasing the depth ceases to be beneficial. In addition, because of the tree-like structure of the network, increasing the depth also increases the training time considerably. Adding one layer will roughly double the number of fitting parameters. It is therefore wise the try multiple depths. For most of the RVEs in this thesis, networks with a depth of $N = 3$, $N = 5$ and $N = 7$ are used. With a total number of fitting parameters of 23, 95 and 383, respectively, the DMN is nevertheless significantly lighter than alternative neural networks. There are model compression techniques like node deletion and subtree merging [1] that can be performed when parts of the network get deactivated as a result of the use of the rectified linear unit.

### 3.2.2. Network regularization

There are several regularization techniques available for neural networks. When using dropout [22], every forward pass a randomly chosen portion of the hidden neurons are (temporarily) omitted. This popular technique causes the network to learn more robust features, since each neuron can not rely on the presence of particular other neurons and is forced to collaborate with many different random subsets of the neurons [40]. Unfortunately, the DMN relies on a fixed shape of the network which makes it impossible to add or remove nodes to a layer arbitrarily. Another popular choice is L2 (or L1) regularization where the network is penalized for high weight values by appending an extra term to the cost function. As already discussed in Section 2.3, in this work a similar regularization term is added to the cost function that prevents the activation fitting parameters $z^j$ from getting too large [1]:

$$C(z, \theta, \lambda) = C_0(z, \theta) + \lambda \left( \sum_j Re(z^j) - \xi \right)^2 \tag{3.2}$$

The hyper-parameter $\xi$ has a clear impact as it determines the average value of the activations $z^j$. When using $\xi = 2^{N-1}$ the average will tend to 0.5, which means that the network is penalised not only for high values of $z^j$, but *also* for values close to zero. When set appropriately, the magnitude of the hyper-parameter $\lambda$ has no influence on the final distribution of the activations, but it can speed up the training process. A too high (or too low) value can cause unwanted early deactivation. Figure 3.7 shows the training and validation history during training with various choices for $\lambda$. In this work $\lambda = 0.001$ is chosen, since it will not slow down the training process significantly, whilst still providing the desired regularization, by constrain the magnitude of the activations. Notable is that in contrast to normal weight decay, all activations $z^j$ are penalized with the same factor, instead of regulating them independently.

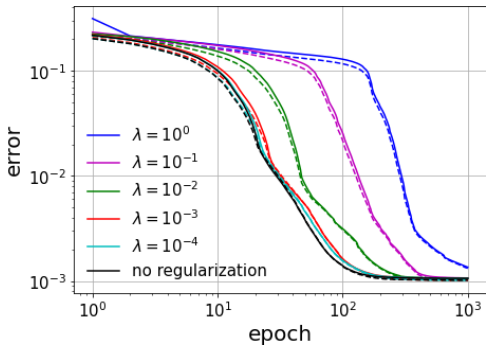



Figure 3.7: Training (solid) and validation (dashed) error during training for different values for the regularization parameter $\lambda$.

This choice for regularization exerts only a constraint on a part of the fitting parameters, namely $z^j$. A regulating element for the remainder of the fitting parameters, the rotations $\theta_i^j$, does not seem to be necessary due to its cycling nature. A rotation of $2k\pi$ for $k \in \mathbb{N}$ is equivalent to $2\pi$.

### 3.2.3. Online step size

When using a trained DMN online it is noticeable that there is a dependency on the step size by which the element is loaded. This is a common feature in some constitutive models, but often negligible. In Figure 3.8 a trained DMN is tested with a uniaxial tension up to $\varepsilon_{11} = 0.01$ with a number of steps ranging from 100 to 100 000 and compared with an FE$^2$ material with a sufficiently small step size. Although in this setup every version seems to predict the behaviour of the RVE quite well, there is an improvement (or at least a difference) with decreasing step size. To confine the computing time, in this thesis a total number of steps of around 1000 is chosen, unless stated otherwise. When applying the DMN on other meshes or with other more complex loading paths, it is wise to experiment with different choices before committing to one.

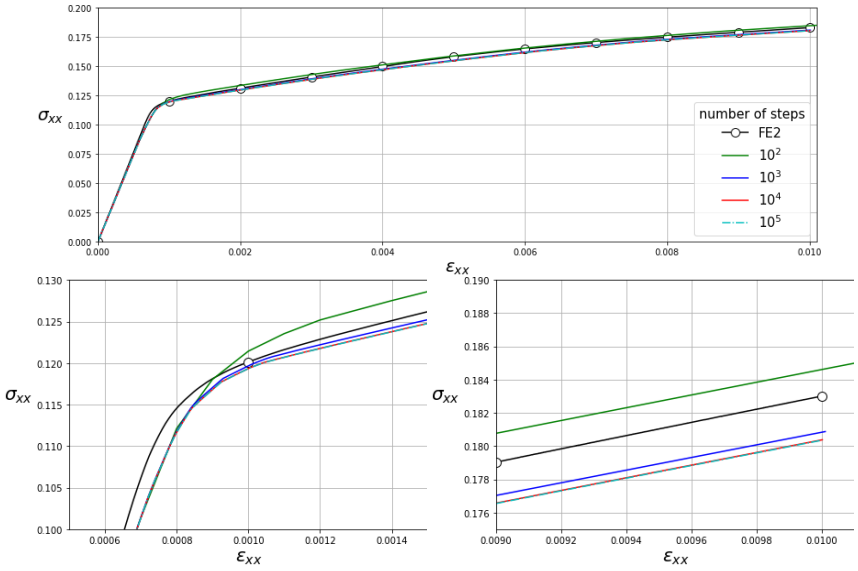



Figure 3.8: Uniaxial tension test using a trained DMN ($N = 7$) with varying step sizes.

## 3.3. Design of Experiments (DoE)

For training the network and finding the values for the fitting parameters that uniquely correspond to a certain RVE (cf. paragraph 3.1.2), a large training dataset is generated. Each sample consists of two input stiffness matrices for the two phases and one output stiffness matrix for the RVE. The first two are generated with random material properties (within a certain input space) and the last one is found using an FE$^2$ model with a single element at the macroscale.

It is not sufficient to only train with the material properties to be used during online prediction. This is mainly due to the fact that the training happens in the elastic regime only. When the yield stress is reached for one of the materials and the response of the RVE becomes nonlinear, the DMN needs to be trained for a wider range of material properties. For the most general case anisotropic stiffness matrices should be used for generating training data, but here both materials are assumed to be orthotropic linear elastic during the sampling. This reduces the sampling size considerably and there still is a high enough degree of anisotropy to accurately predict the behaviour of the the material models used in the online stage. It should be noted that orthotropy is assumed during training even though the phases are actually isotropic. The stiffness matrices of the two phases in Mandel notation can be written as:

$$\boldsymbol{D}^{pi} = \begin{bmatrix} E_{11}^{pi} x & \nu E_{11}^{pi} x & \\ & E_{22}^{pi} x & \\ \text{sym} & & 2G_{12}^{pi} \end{bmatrix} \qquad \text{with} \qquad x = \frac{1}{1 - \nu^2 \frac{E_{11}^{pi}}{E_{22}^{pi}}} \tag{3.3}$$

Each material has four independent design variables: $E_{11}$, $E_{22}$, $\nu$ and $G$. The Young's moduli of each phase are first randomly sampled. Because during training the actual values are less important than the ratio between them, the sampling adheres to the following boundaries:

$$\begin{aligned} &E_{11}^{p1} E_{22}^{p1} = 1, \quad \log_{10}(E_{11}^{p2} E_{22}^{p2}) \in U[-4,4], \\ &\log_{10}(E_{22}^{pi} / E_{11}^{pi}) \in U[-1,1] \end{aligned} \tag{3.4}$$

To guarantee all the randomly sampled matrices are positive definite the shear modulus and Poisson's ratio of each phase are sampled as:

$$\frac{G_{12}^{pi}}{\sqrt{E_{11}^{pi} E_{22}^{pi}}} \in U[0.25, 0.5], \quad \frac{\nu_{12}^{pi}}{\sqrt{E_{22}^{pi} / E_{11}^{pi}}} \in U[0.3, 0.7] \tag{3.5}$$

For each sample there are seven variables that need to be fixed. If for each variable a uniformly spaced set is drawn and the training data in generated through exhaustive sampling, then the sample size will grow too large or the sample space will not be sampled extensively enough. As a compromise the variable that determines the ratio between the Young's moduli of the two materials $\left(\log_{10}(E_{11}^{p2} E_{22}^{p2}) \in U[-4,4]\right)$ is sampled from a uniform distribution and the other six variables are chosen randomly. As discussed in paragraph 3.3.3 a dataset of 1000 samples this method is enough to cover the whole parameter space.

It is challenging to find the ideal way of creating robust training data. The network needs enough examples to train on, but it does not necessarily need to see all the possibilities. Because the pattern recognition follows the homogenization operations in each building block, training on a subspace could be sufficient. The sample space is by definition different from the space that is used online. Intuition based on experience suggests that the aforementioned sampling method yields good results. To construct a respectable sampling scheme it turns out that not all the seven variables are equally important. Figure 3.9 shows the uniaxial tension test with seven trained DMNs, compared with $FE^2$. For each case the training data is sampled differently. Six of the seven variables are kept constant and the remaining variable is sampled uniformly within the ranges defined above. It seems that an extensively sampled ratio between the Young's moduli of the two materials is the most decisive, although rigorous claims are hard to make.

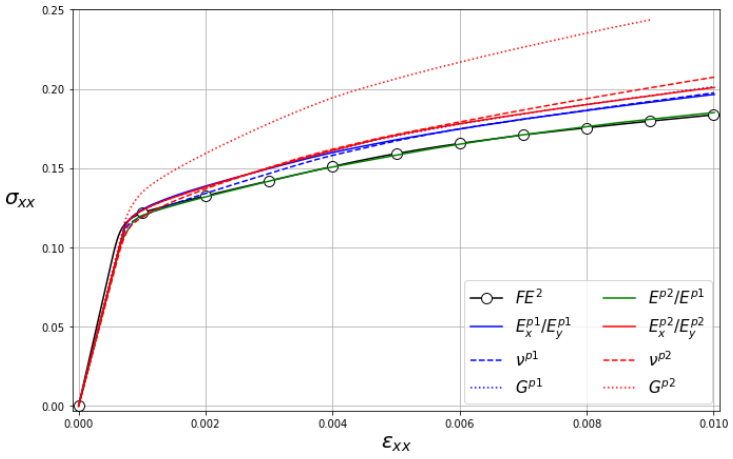
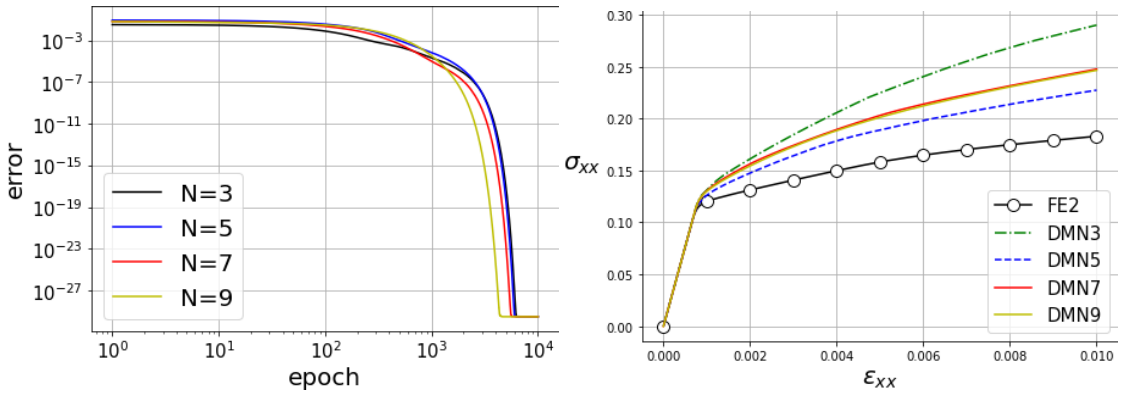



Figure 3.9: Uniaxial tension test to compare importance of the sampling variables.

For comparison, the network is also trained with only one sample, consisting of precisely the two stiffness matrices that are used online. The trained DMN is then used for online prediction. Intuitively one would expect a low training error corresponds to a well trained neural network. The very low error indeed corresponds to the excellent prediction while both phases still are in the elastic regime, but the plastic prediction is below par. The reasoning behind this lies in the fact that the network is not merely trained for mapping the strain to the stress, but is trained to construct a stiffness matrix (cf. Figure 1.15). It does not need to see all possible scenarios it could encounter online, but it does need to see enough distinctive examples, where the connection to realistic scenarios is of secondary importance. This characteristic is what makes the DMN so powerful, yet difficult to control.




(a) Training error using one sample.

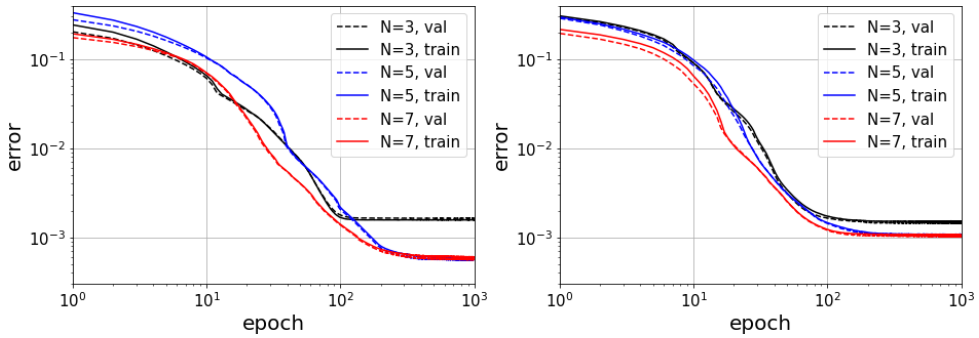
(b) Uniaxial tension test using a DMN trained with one sample.

Figure 3.10: Training error during training with only one sample (a) and online prediction (b).

### 3.3.1. Orthotropic versus isotropic samples

The need for expanding the sample space to orthotropic elastic, even though in the online phase only isotropic materials are used, is illustrated below. In Figure 3.11 the training and validation errors are shown for two sample spaces, for network depths of $N = 3,5,7$. The orthotropic sample space is formed using the method above and the stiffness matrices of the two materials for the isotropic sample space are formed using the boundaries from Eq. (3.6). The Young's modulus of material 1 is set to 1, the Young's modulus of material 2 is generated by a uniform distribution and the two Poisson's ratios are randomly drawn. For both methods a sample size of 1000 is used, of which 800 for training and 200 for validation.

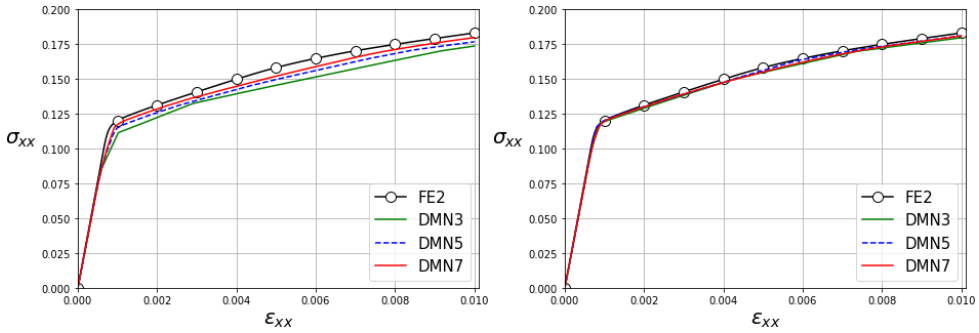
$$\begin{aligned} &E^{p1} = 1, \qquad \nu^{p1} \in [0.005, 0.495] \\ &\log_{10}(E^{p2}) \in U[-2,2], \qquad \nu^{p2} \in [0.005, 0.495] \end{aligned} \tag{3.6}$$




(a) Isotropic. (b) Orthotropic.

Figure 3.11: Histories of average training errors and validation errors with a sample space limited to (a) isotropic matrices and (b) orthotropic elastic matrices.

In standard neural networks the validation error yields a considerable meaning about the reliability of the network. This is a less reliable metric for a DMN, since it is only trained in the elastic regime. The isotropic sample space seems to generate better results since the error is lower, but when using the trained network in online prediction the orthotropic sample space produces better results. In Figure 3.12 both networks are tested with uniaxial tension. It is clear form the plots that training the network with only isotropic matrices is insufficient to capture the plastic behaviour.

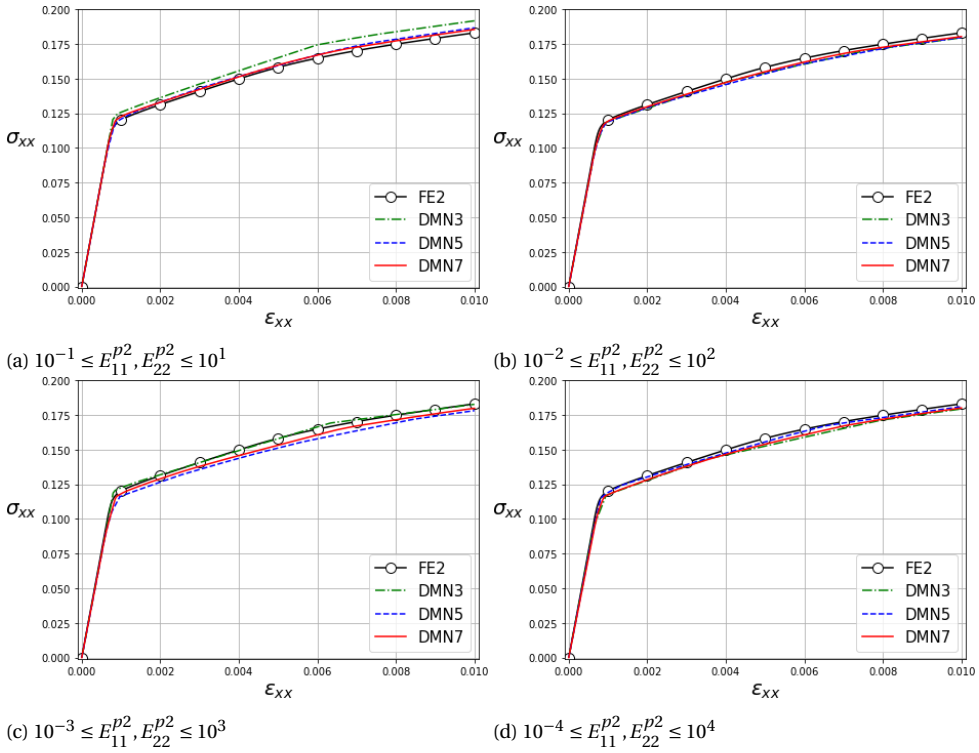



(a) Isotropic. (b) Orthotropic.

Figure 3.12: Comparison between DMNs trained with an (a) isotropic and (b) orthotropic sample space.

### 3.3.2. Range of the training samples

The variable that determines the ratio between the Young's moduli of the two materials, $\log_{10}(E_{11}^{p2}E_{22}^{p2})$, is for each sample in the training data drawn from a uniform distribution between $-4$ and 4. To clarify, in the equation the log is taken of the product of the two Young's moduli. That means that $U[-4,4]$ gives a maximum ratio between the phases of $\sqrt{10^4} = 100$. To justify this range, the network is trained four times with increasing sample ranges. In order to be able to make a consistent comparison, the sample density is kept the same. Therefore the sample spaces $U[10^{-4},10^4]$, $U[10^{-3},10^3]$, $U[10^{-2},10^2]$ and $U[10^{-1},10^1]$ have a sample size of 2000, 1500, 1000 and 500, respectively.




(a) $10^{-1} \leq E_{11}^{p2}, E_{22}^{p2} \leq 10^{1}$ (b) $10^{-2} \leq E_{11}^{p2}, E_{22}^{p2} \leq 10^{2}$

(c) $10^{-3} \leq E_{11}^{p2}, E_{22}^{p2} \leq 10^{3}$ (d) $10^{-4} \leq E_{11}^{p2}, E_{22}^{p2} \leq 10^{4}$

Figure 3.13: Influence of the range in the ratio between the Young's moduli of the two materials when creating training data.

All four options give good predictions, but from these results a choice for $10^{-2} \leq E_{11}^{p2}, E_{22}^{p2} \leq 10^{2}$ is understandable. It is however good to be on the safe side since in practice more complex loading paths will be used. The ratio between the elastic Young's moduli of the two phases that is used here is 5. If the network is used with more extreme differences between the properties of the two material phases, a larger sample space could be needed. Increasing the sample space will subsequently require other measures like the use of a deeper network, because, as seen in Figure 3.14, the training process can become unstable. Even though the sample density is kept constant, a larger sample space causes a cutback in accuracy, as expected.

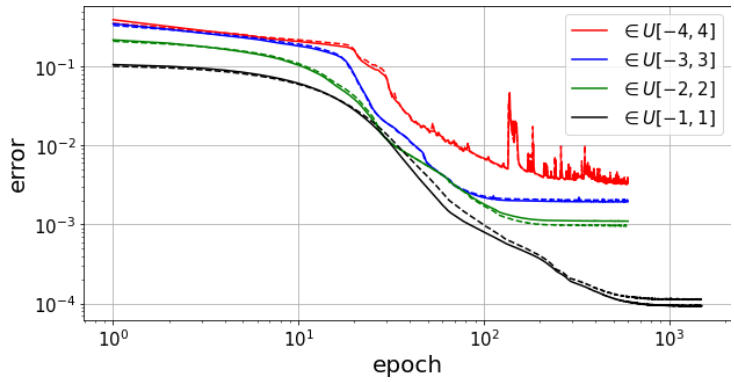
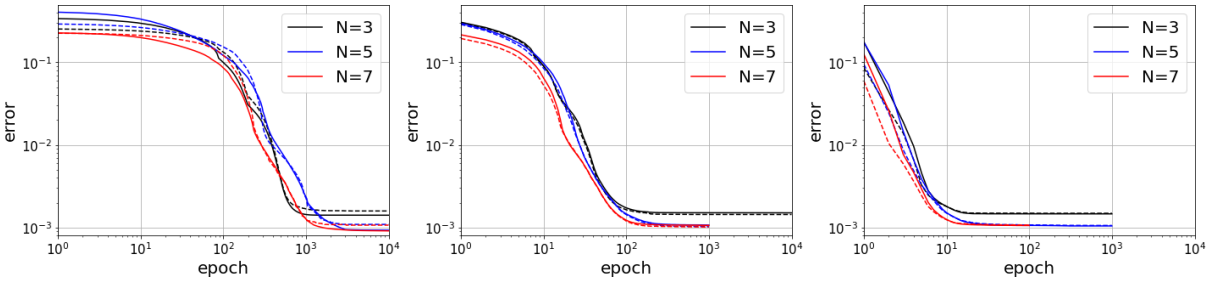



Figure 3.14: Training (solid) and validation (dashed) error during training with depth $N = 7$ for different limits to the ratio of the Young's moduli between the two materials in the training samples.

### 3.3.3. Sample size

In Figure 3.15 the training and validation error during training is shown for training with 100, 1000 and 10000 training samples (of which 1/5th is set aside as validation set). The network with a larger sample size needs less epochs, because the fitting parameters are updated more often per epoch; after every minibatch of 32 samples. Increasing the sample size reduces overfitting and ensures that the whole sample space is covered. A sufficiently large sample size is especially important when the samples are generated partially randomly. A

sample size of 1000 is chosen in this thesis.




(a) 100 samples. (b) 1000 samples. (c) 10000 samples.

Figure 3.15: Training (solid) and validation (dashed) error with increasing sample size.

# 4

# Results

The fidelity of the material network will now be assessed by using it as a surrogate model for a more intricate RVE with realistic phase properties. In the next section linear elastic response high contrast of phase properties is evaluated. After that the ability to extrapolate nonlinear plastic behaviour is put to the test, with the emphasis on unloading.

## 4.1. Elastic response of a realistic RVE

From now on a more detailed RVE is used, seen in Figure 4.1. With the increase of elements (from 828 to 7088), the increase in computing speed becomes truly evident. The micro-heterogenous material consists of fibers embedded inside the matrix. First the elastic behaviour of the RVE is analyzed. A small uniaxial strain is imposed, to find the stiffness matrix.

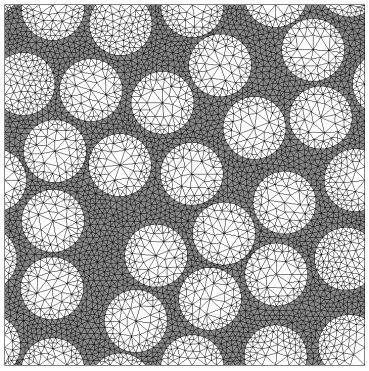


Figure 4.1: Mesh of a two-phased RVE resembling a fiber enforced material with fiber volume fraction of 50%.

When training the network, the training and validation errors give an average of the error. The average should be below some acceptable threshold, but if too many single samples give an error which is too high, the network cannot be trusted. To give more insight whether the error of the DMN is constant or that it has unexpected variability, a trained network is compared with an $FE^2$ model for four different situations. Four batches of 1000 isotropic samples are created with the following material properties.

$$
\begin{aligned}
&E^{p1} = 1, \qquad \nu^{p1} \in [0.005, 0.495], \qquad \nu^{p2} \in [0.005, 0.495] \\
&\text{batch 1: } \log_{10}(E^{p2}) \in U[-1,1] \\
&\text{batch 2: } \log_{10}(E^{p2}) \in U[-2,2] \\
&\text{batch 3: } \log_{10}(E^{p2}) \in U[-4,4] \\
&\text{batch 4: } \log_{10}(E^{p2}) \in U[-8,8]
\end{aligned}
\tag{4.1}
$$

The Young's modulus of phase 1 is 1 to remove the redundancy due to the scaling effect, the Poisson's ratios are sampled randomly and the Young's modulus of phase 2 is drawn from a (logarithmic) uniform distribution with increasing range for each batch.

In Figure 4.2 the four batches are compared using a histogram of all the 1000 errors. There are some outliers in every batch, but there is a clear trend visible that show that higher errors are less frequent. In Figure 4.3 the same errors are plotted, with on the y-axis the error and on the x-axis the corresponding ratio $E^{p2}/E^{p1}$. Here it can be concluded that the larger errors correspond to extreme cases and crucially, the unacceptable errors only occur when the fibers (phase 2) are softer then the matrix (phase 1), which is generally not the case. The high error for soft fibers appears to emanate from the fact that the norm of the surrogate stiffness matrix is consistently too low, and when using the cost function Eq. (2.7) this results in an error of 0.5. Although an explanation for this behaviour still warrants further investigation, fibers are traditionally stronger. The extrapolation to relatively tougher fibers seems to work better as the error stays below 0.025.

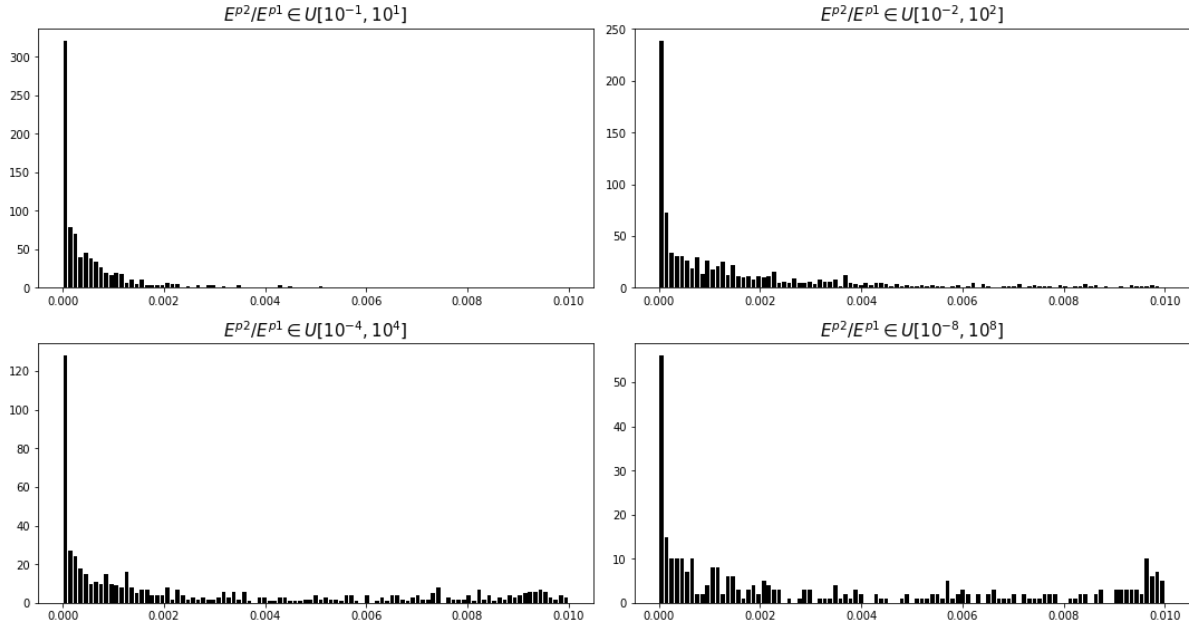



Figure 4.2: Histograms of the error of the four batches.

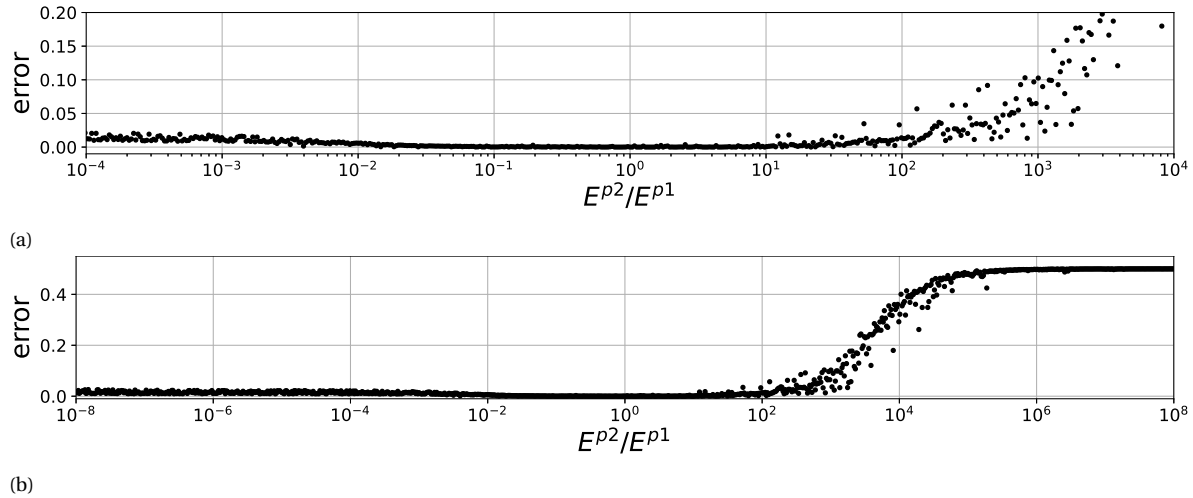



Figure 4.3: Error distribution with on the x-axis the stiffness ratio; stiff fibers left, soft fibers right.

## 4.2. Nonlinear plasticity with small strain

We now fix the material properties used to make online predictions. The fibers stay purely elastic with $E^{p1} = 74000$ and $\nu^{p1} = 0.2$. For the matrix, the pressure-dependent elastoplastic model proposed by Melro et al. [38] is used with $E^{p2} = 3760$, $\nu^{p2} = 0.3$, a plastic Poissons's ratio $\nu^{p2}_{pl} = 0.39$ and hardening behavior given by Figure 4.4. First yield stress envelopes are created to investigate the general behaviour of the DMN in a range of loading angles. After that a number of loading paths are considered containing loading and unloading at different stress/strain levels.

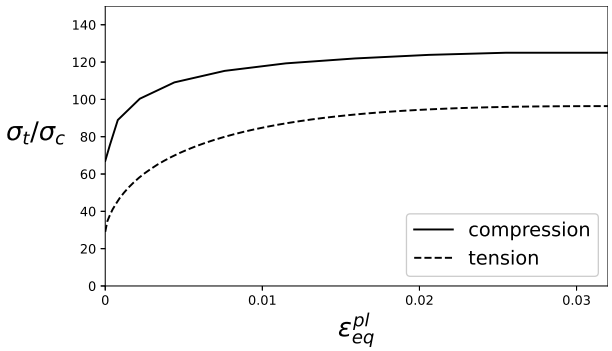
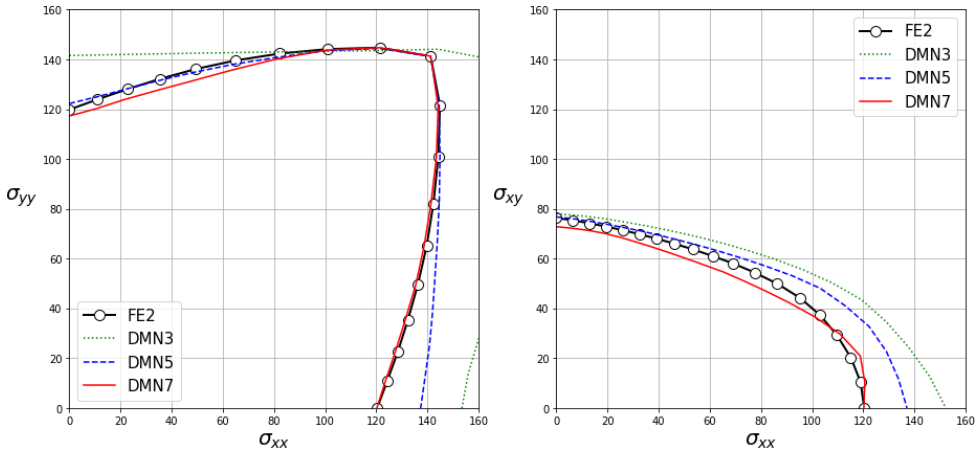



Figure 4.4: Input hardening curves for micromechanical plasticity model. [41]

### 4.2.1. Yield stress envelope

A yield stress envelope is a collection of stress combinations where the equivalent plastic strain corresponds to a certain value. This way it shows the material response in a wide range of loading combinations. When dealing with plastic behaviour, such envelopes are commonly used by plotting the final stress levels for different stress ratios. For the two yield stress envelopes in Figure 4.5, a simplified approach is adopted and the strain tuples are obtained from when $\sqrt{\varepsilon^2_{xx} + \varepsilon^2_{yy}} = 0.04$ or $\sqrt{\varepsilon^2_{xx} + \varepsilon^2_{xy}} = 0.04$, which lies in the perfect plasticity regime, allowing us to disregard the actual equivalent plastic strain to which the envelope is associated. Steps in the stress ratio are made of $\theta = 5°$, where $\tan\theta = \frac{\sigma_{xx}}{\sigma_{yy}}$.




(a) $\varepsilon_{xx} - \varepsilon_{yy}$.

(b) $\varepsilon_{xx} - \varepsilon_{xy}$.

Figure 4.5: Yield stress envelopes.

In Figure 4.6 the yield stress envelope for all four quadrants is shown for $FE^2$ and for two DMNs with a depth of $N = 7$ and $N = 9$. All the points where no perfect plasticity was reached are omitted. In the third quadrant the DMN could find a solution (albeit not entirely accurate given that it does not form a smooth curve), where the $FE^2$ could not. A second curve is formed by a collection of points where the norm is equal to 0.01, where plasticity is started, but not yet final. Here it is observed that the DMN is persistently overestimating.

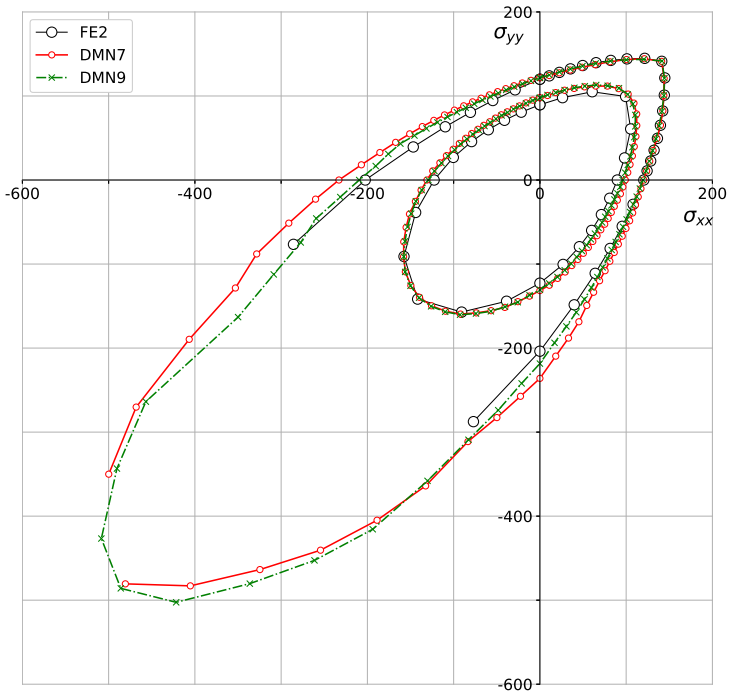
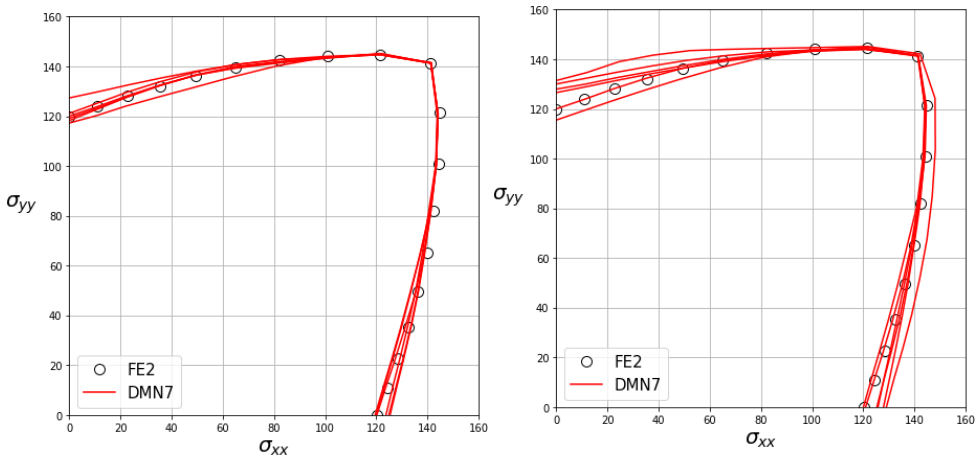



Figure 4.6: Yield stress envelope with all four quadrants, for $\sqrt{\varepsilon_{xx}^2 + \varepsilon_{yy}^2} = 0.01$ & $0.04$.

A network depth lower than $N = 7$ is clearly not enough to guarantee a good approximation. For ratios near $\theta = 45°$ the biaxial tension test seems to work well, but near $\theta = 0°$ and $\theta = 90°$ a perfect fit is not there. Every result generated using a neural network has an uncertainty originating from the choices for the initial fitting parameters $(z^j, \theta_i^j)$ and from the used training data. Even though local minima of large networks exhibit the structure where high-quality local minima lie close to the global minimum [42], different trained DMNs can still exhibit variety in precision. In Figure 4.7a multiple DMNs of depth $N = 7$ are trained with different training (and validation) sets, but with the same initial values for the fitting parameters, and Figure 4.7b shows multiple DMNs trained with the same training data, but different initial fitting parameters. Especially near uniaxial tension they behave differently. This has probably to do with the fact that this network has trouble processing the effect of the Poisson contraction online. During training an optimum is sought in the multidimensional surface of the cost function. Depending on the starting situation the training algorithm can land in one of the many local minima and it is not even clear if the global minimum gives in fact the best trained network, since the DMN is used in a different way when using it to extrapolate online. Although each displayed version of the network shows fairly good results, it is possible to optimize the competence with multiple realizations of the network with different sets of initial parameters. The DMN with the smallest error when using it to make a yield envelope will be selected as the surrogate for the RVE. Other tests can be done instead, but crucially it must be a test where the network is used in a variety of loading scenarios. When choosing the most desirable result with normal neural networks, it would suffice to select the one with the minimum training or testing error, but that is not enough for this network since the validation error is merely an indication.




(a) Different training data.

(b) Different initial fitting parameters $(z^j, \theta_i^j)$.

Figure 4.7: Yield stress envelope biaxial loading for six differently trained DMNs.

### 4.2.2. Extension to the training process

An important extension to the training process can be made here. As stated before, during the training of the network, the training and validation errors are minimized. Unlike for conventional neural networks, this error does not necessarily identify the best network when running the training procedure for different starting situations (initial fitting parameters or training set), cf. Figure 3.10. This shortcoming can be alleviated by inspecting the yield stress envelopes. The training history of all six DMNs from Figure 4.7a are presented in Figure 4.8 (the labels are omitted for clarity). The six networks are trained with the same initial fitting parameters, but with different training data. They all settle around the same error, and the observation that the validation error (dashed) is occasionally higher than the training error (solid) does not seem the have a correlation with the accuracy of the network. In algorithm 5 the extension to the training procedure from Liu et al. [1] is summarized. In this thesis the envelope evaluation is done by visual inspection and for the upcoming tests the DMN labeled *c* is used.

**Algorithm 5:** Extention of the training method.

1 **for** *n in N* **do**
2 Train network *n* for sufficient number of epochs.
3 Use the fitting parameters $(z^j, \theta_i^j)$ to make one envelope.
4 Assess the accuracy by applying the mean square error.

**Output:** Choose the network with the lowest error.

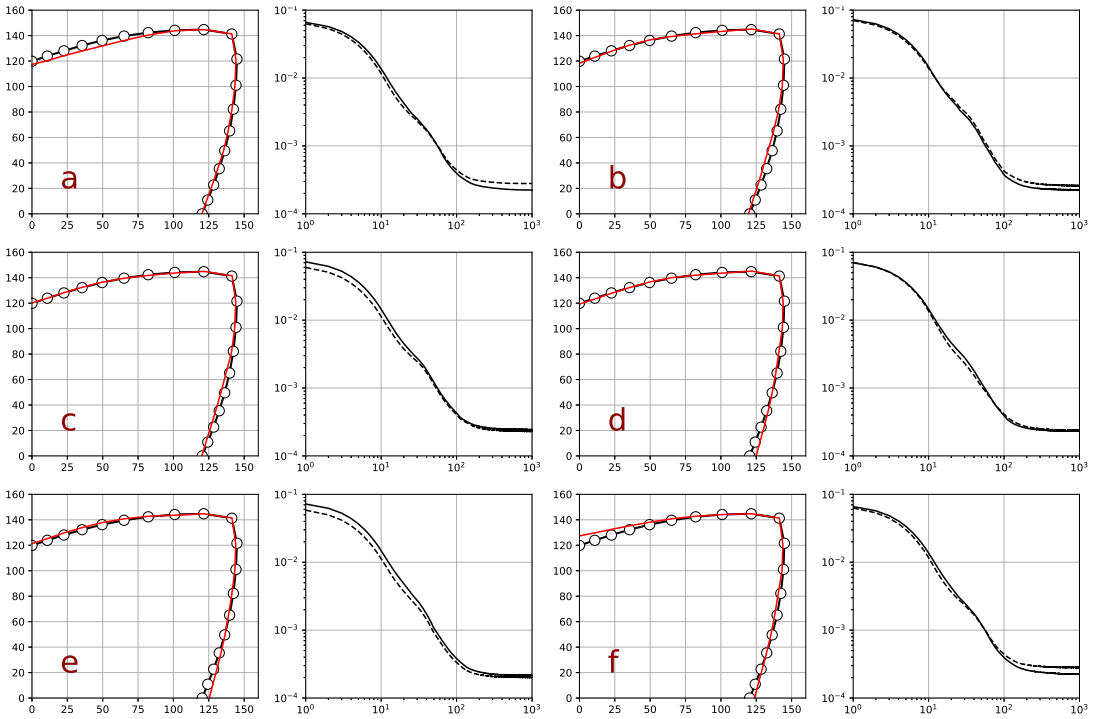



Figure 4.8: Six network of depth $N = 7$ are trained with the same $(z^j, \theta_i^j)$, but with training sets generated with a different random seed.

### 4.2.3. Different loading paths

In this section the trained DMN is used to predict the behaviour of the RVE for different loading situations, mainly focusing on the aspect of unloading. At this point it is worth recalling the main intention for exploring this new material network. Other surrogate models based on neural networks are fundamentally weak regarding unloading, especially when the loading path is outside of the training space. An FNN (cf. paragraph 1.4) is incapable of capturing unloading and an RNN has a number of drawbacks. In Figure 4.9 the RVE is loaded up to the ultimate yield stress and then brought back to zero total strain. The loading-unloading behaviour can be well captured, especially considering the network has never seen constitutive relations like this before. Also the loading path in Figure 4.10 where the element is unloaded an reloaded in three different stages is easily handled with. When taking a closer look at both figures, it can be seen that DMN3 and DMN5 still have a positive slope and did not reach perfect plasticity, which can accumulate. The DMN7 does obtain perfect plasticity, but overshoots initially as well. That correlates to the the inner envelope of Figure 4.6, where the DMN presents a higher stress for a strain of 0.01. Interestingly, the three unloading intermezzos in Figure 4.10 do not seem to interfere with the global stress path.

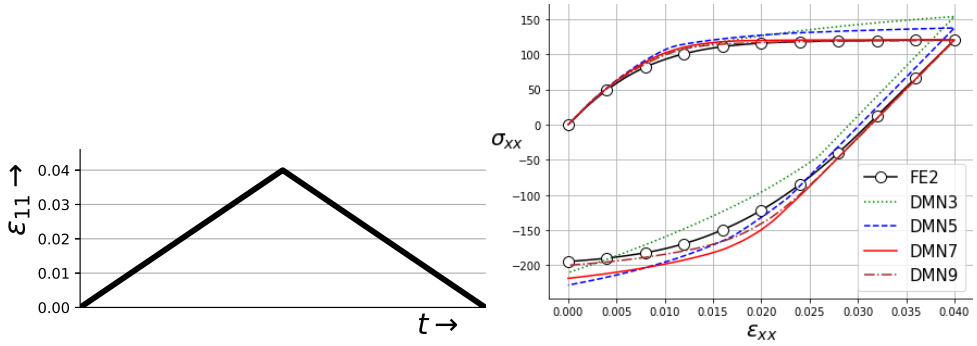



Figure 4.9: Uniaxial tension with single unloading.

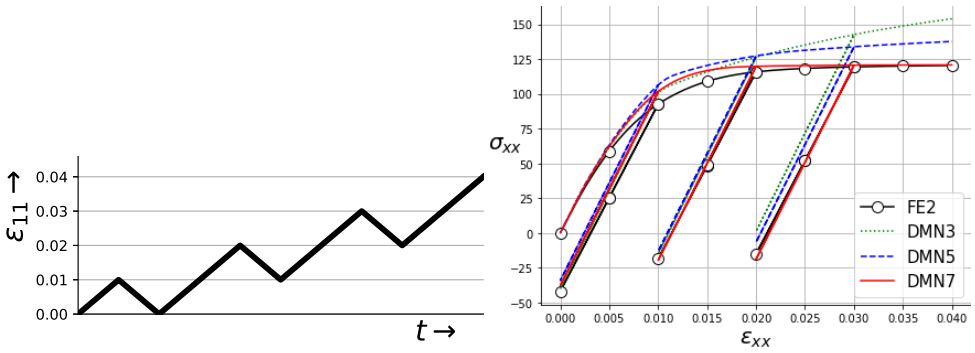



Figure 4.10: Uniaxial tension with loading and unloading in stages.

In Figure 4.11 a more complex loading path is examined. As a consequence of the uninformative training procedure of this network, it is not always clear in advance how deep the network should be for a certain loading case. When comparing the curves for $N = 3, 5, 7$ it can be concluded that increasing the depth of the network to $N = 9$ would be beneficial. Also the online step size, as discussed in section 3.2.3, could be the cause of an inadequate prediction. Here the step size is small enough. Indeed, DMN9 seems to be a better fit overall, other than the small sidestep in the shear graph, an unforeseen behaviour also present when the network with depth $N = 9$ was retrained.

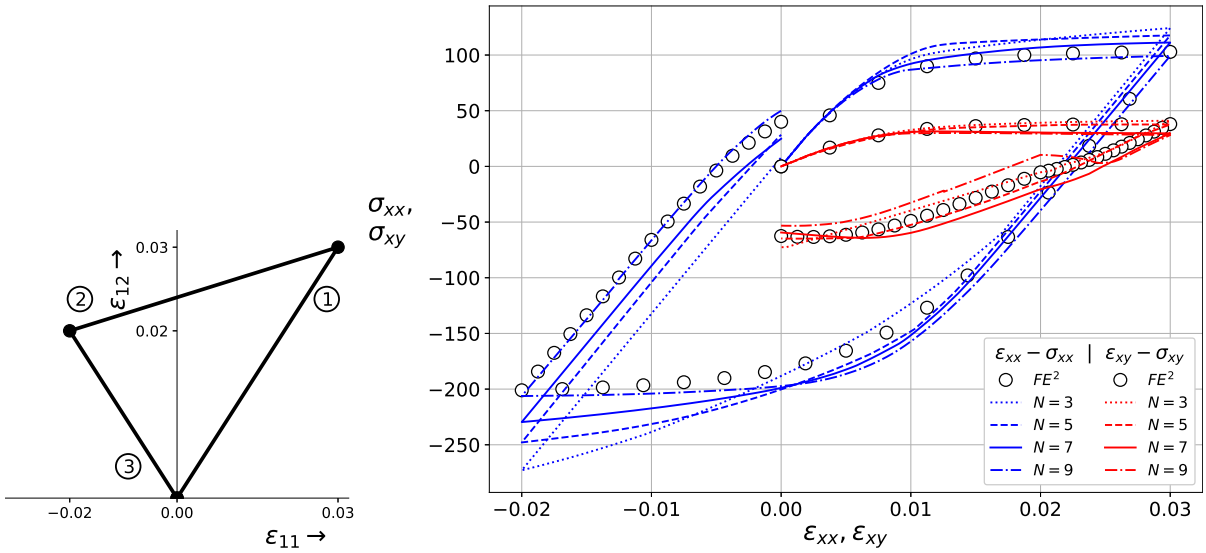



Figure 4.11: Loading path combining tension and shear.

To really push the envelope an even more challenging loading path is considered, for academic purposes. In Figure 4.12 two similar loading paths for uniaxial tension are shown. The oscillating sinusoidal, that attempts to resemble the noisiness that sometimes occurs, has the same maximum strain as the sinusoidal. Again, the surrogate models in Figure 4.13 satisfactorily follow the requested path, but the oscillation does hamper the DMN slightly.

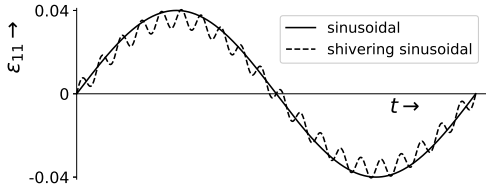
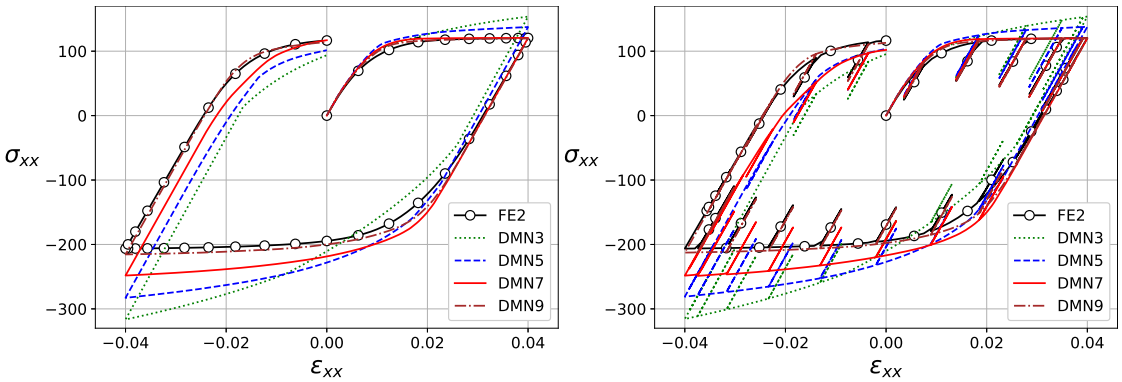



Figure 4.12: Two similar loading paths.



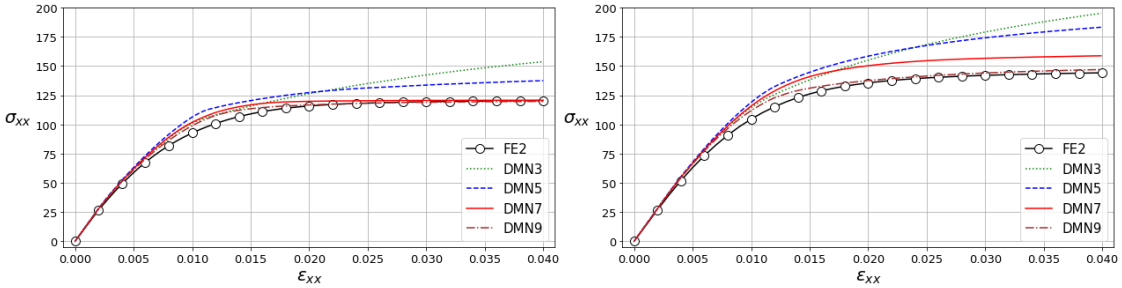

(a) Uniaxial sinusoidal tension.

(b) Uniaxial shivering sinusoidal tension.

Figure 4.13: Two stress-strain graphs with related loading paths.

### 4.2.4. Different hardening laws

For completeness, the performance of the DMN is also tested with other hardening laws. If the compression hardening is set equal to tension hardening (cf. Figure 4.4) the proficiency of the DMN is apparently reduced, when looking at Figure 4.14. Attaining perfect plasticity is only possible with a depth of $N = 9$.

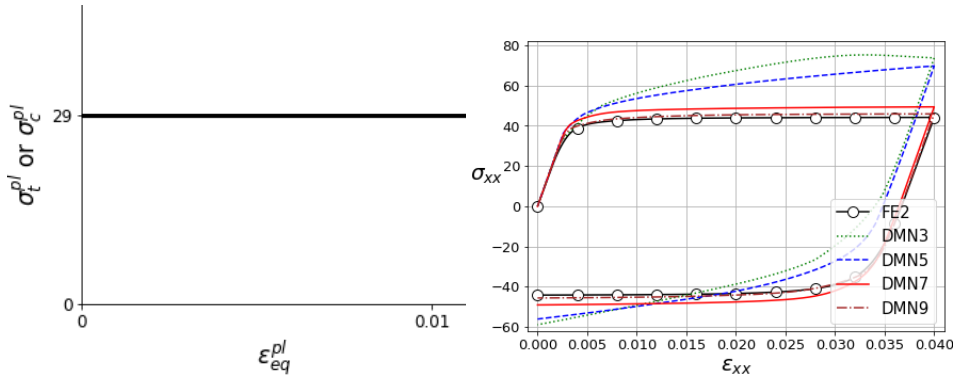



(a) Hardening law as in Figure 4.4.

(b) Compression hardening is set equal to tension hardening.

Figure 4.14: Uniaxial tension test.

If the non-fiber material in the RVE has a rigid perfectly plastic model it is still possible to adequately represent the behaviour of the RVE, as seen in Figure 4.15. As seen before (in the discussion about Figure 4.9) a DMN of any depth $N$ overreaches when transitioning to plasticity. However, when the transition is too abrupt there is no room to resolve this issue. A sharp transition in the strain path on the other hand is without problems, in contrast to the RNNs of Figure 1.11.




(a) Rigid plasticity.

(b) Stress-strain curve.

Figure 4.15: Uniaxial tension with loading and unloading.

In Figure 4.16 the loading path of Figure 4.9 is repeated, but for J2-plasticity. The plastic Poisson's ratio of the matrix is set to $\nu_{pl}^{p2} = 0.49$ and the compression hardening curve is set equal to the tension hardening curve.

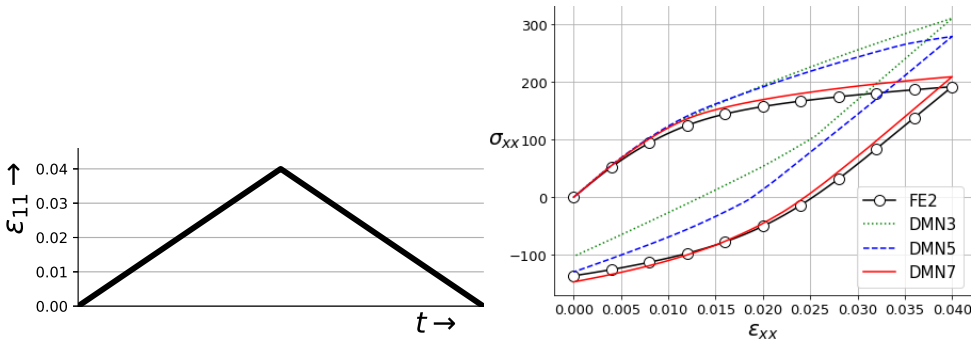



Figure 4.16: Single uniaxial loading with unloading, with J2-plasticity.

### 4.2.5. Computing time

To give an indication of the advancement in time, the RVE in Figure 4.17 is once again loaded. The horizontal strain is brought to $\varepsilon_{xx} = 0.04$ in 400 steps of $\Delta\varepsilon_{xx} = 0.0001$. For this fiber enforced element with 7088 elements the FE simulation with a single macroscopic integration point took $791s$. The DMN surrogate with a depth of $N = 3$ and $N = 5$ were done within a second, $3s$ for $N = 7$ and $14s$ for $N = 9$. For more complex loading cases the computational time of the DMN could increase, as the constitutive relations of the nodes in the bottom layer that correspond to the plastic material need more iterations to be solved, but this also applies to the FE simulation. This time increase with a factor of about 250 (for $N = 7$) is dependent on the size of the RVE. Increasing the number of elements in the RVE also increases the computational time of a full order model, but the speed of a trained surrogate DMN is independent of the original complexity of the RVE. The same holds for training the network. Generating the training data, however, is dependent on the number of elements, albeit with a smaller order of magnitude. During the formation of one training sample the full RVE needs only a single evaluation, in the elastic regime. During an FE simulation the micromodel has to be evaluated each load step, which takes even more time with nonlinear behaviour. This advance in time will be even more pronounced for 3D RVEs.

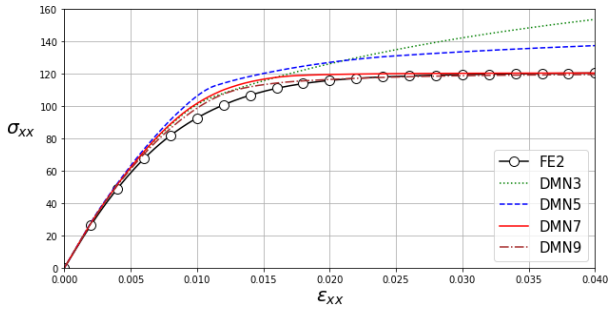



Figure 4.17: Uniaxial tension test.

The computational time is linearly proportional with the number of fitting parameters $N_{z,\theta}$ in the network. The relation with the depth $N$ is $N_{z,\theta} = 3 \cdot 2^N - 1$. In Figure 4.18 the computational time is plotted against $N_{z,\theta}$ for a depth of $N = [3,5,7,9,10]$. To make the time more measurable, $\varepsilon_{xx} = 0.04$ is reached with 4000 steps and a DMN of depth $N = 10$ is also included.

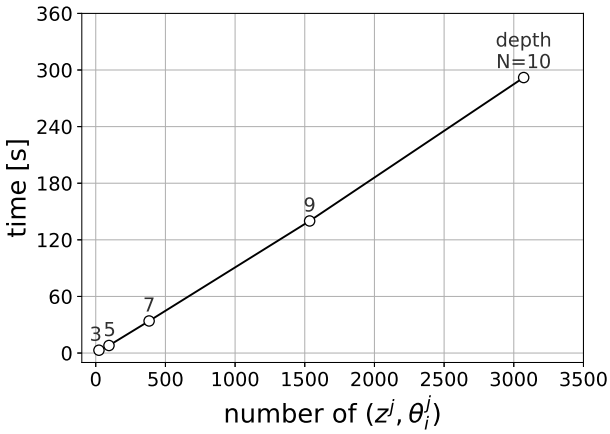



Figure 4.18: Computational time versus number of fitting parameters.

# 5

# Conclusions and future work

A novel multiscale surrogate model, proposed by Liu et al. [1], is studied with promising features that open interesting new insights in the union of concurrent multiscale modeling and machine learning. The Deep Material Network (DMN) is a data-driven multiscale material modeling method for modeling general heterogeneous materials with nonlinear constitutive behaviour. With the use of machine learning techniques the network is trained with offline data solely consisting of stiffness matrices in the elastic regime after which it can be used to predict highly nonlinear behaviours in the online stage.

## 5.1. Conclusion

The main target in this thesis was to investigate the newly proposed DMN and its capabilities to capture loading-unloading behaviour of a multiscale model. Other surrogate models that use machine learning techniques to fill the gap between the stress and the strain of the RVE do this by fully replacing the constitutive relation with a neural network, and thereby discarding the physics that is embedded. What makes the DMN fundamentally stand out is that it instead (as already visually displayed in Figure 1.15) replaces something different, namely the coupling of the two length scales by finding an alternative way for the construction of the stiffness matrix of the RVE. There are still actual (simple) constitutive models being evaluated at a certain stage and the network is trained to convert their outcome to a tangent stiffness matrix of the macroscopic integration point. That way it is possible to circumvent the need for online scale coupling, while the relation between the scales is preserved. Practical ingredients related to the two materials like loading-unloading conditions and rate dependence, are explicitly conserved.

The DMN was indeed capable to accurately translate microscopic behaviour in an RVE to the macroscopic level and it can certainly operate as a surrogate constitutive model. The extrapolating abilities to unknown material properties and unknown loading paths look remarkable and the accuracy seems to be predominantly determined by one's devoted training time and the depth of the network. In chapter 4 the arrangement of the training procedure is treated. This is not a straightforward process since the tree-like structure of the network is built up of sophisticated building blocks with intricate homogenization processes. Where conventional neural networks have rather elegant relations between the layers with simple derivatives, the backward and forward propagation operations of the DMN are considerably more complex. This is indeed the result of the fact that the embedding of the physics planted in the creation of these building blocks.

Determining the optimal settings for the (hyper) parameters of the DMN is an intricate affair. The main reason for this is that, even though the total procedure is by its very nature solely based on real homogenizations of constitutive relations, logic behind the network is lost. For generating the training data choices must be made about the method of sampling and about the range of sample space. In paragraph 3.3.1 is demonstrated that the expension of the sample space to the use of orthotropic material behaviour yields an advantage, even if only isotropic materials are used online.

When employing this material network for extrapolating purposes, it is advised to first do a small performance study with different depths, based on the results gained in chapter 4, where the parameters of the DMN are optimized. As an addition to the training procedure described by Liu [1] this thesis proposes in 4.2.2 an extra

test where the yield stress envelope is utilized to compare separate DMN trained independently of each other. When training conventional neural networks analysing the training and validation history can almost exclusively determine the accuracy of the network. However, for a DMN a low training error is not the only indicator of a well trained network.

Other surrogate models that use model order reduction broadly still have the capacity to adapt to unforeseen loading situations, but are generally slower than surrogate models that use machine learning. Conventional neural networks on the other hand are generally quick, but only capable of handling situations that match situations that it is trained for. The DMN appears to combine the positive sides of both fields. For all the examples the effectiveness of DMN is demonstrated on representing a complex RVE microstructures and its accuracy of predicting highly nonlinear behaviors in the online stage. When assigning complex unloading situations the DMN does not seem to encounter difficulties.

## 5.2. Recommendations for future work

This thesis offers insight in a promising new way of developing physics informed neural networks. There are however some aspect that are worth revisiting for a better understanding of the convoluted workings that are happening in this "transparent black box".

**Deeper understanding of the training process**
Even though a comprehensive investigation is done in section 3.3 finding an optimized method for generating the training data, there is still potential for improvement. A deeper understanding of the training process can result in less heuristic choices for a proper data sampling strategy. Consequently smarter machine learning methods can optimize offline training even further, by for example preventing early deactivation of activation parameters by modifying the regularization component of the cost function.

**Extension to the training process**
One major advantage of this Deep Material Network is that the dataset used for training is sampled exclusively from linear elastic RVE data. During training the error is minimized, but as demonstrated in paragraph 4.2.2, the history of the training and validation error only gives limited information. If the training process would also include testing the network in the plastic regime, that could arguably be beneficial. The extension to the training process proposed includes information collected from yield stress envelopes to evaluate the fidelity of the DMN. Finding an extension was originally not one of the objectives of this thesis, so not all possibilities are investigated. A more in-depth investigation could be done to expand the training process. This would mean, however, that the valuable property that a DMN only needs linear elastic data to represent an RVE needs to be relaxed, with the associated consequences like a bias of the network towards the cases seen in training.

**Challenge the network**
More complex nonlinearities, history dependencies, RVE topologies and hardening laws can be investigated. During the duration of this project it has always been tried to find a situation where some of the claims do not hold. It would be interesting to look for the boundaries of the possibilities.

**Multiple macroscopic integration points**
In this thesis the DMN is compared with the $FE^2$ method on one macroscopic element with a single integration point. The strength of the DMN can be tested with different macroscopic variations, where it is expected that the DMN will act as any other material model. Furthermore, the time comparison of of section 4.2.5 can be further elaborated.

# Bibliography


[1] Zeliang Liu, C. T. Wu, and M. Koishi. "A Deep Material Network for Multiscale Topology Learning and Accelerated Nonlinear Modeling of Heterogeneous Materials." In: *CoRR* abs/1807.09829 (2018). URL: http://dblp.uni-trier.de/db/journals/corr/corr1807.html#abs-1807-09829.

[2] V. Kouznetsova, M. G. D. Geers, and W. A. M. Brekelmans. "Multi-scale constitutive modelling of heterogeneous materials with a gradient-enhanced computational homogenization scheme". In: *International Journal for Numerical Methods in Engineering* 54.8 (2002), pp. 1235–1260. DOI: 10.1002/nme.541. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.541. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.541.

[3] Frédéric Feyel. "A multilevel finite element method (FE2) to describe the response of highly non-linear structures using generalized continua". In: *Computer Methods in Applied Mechanics and Engineering* 192.28 (2003). Multiscale Computational Mechanics for Materials and Structures, pp. 3233–3244. ISSN: 0045-7825. DOI: https://doi.org/10.1016/S0045-7825(03)00348-7. URL: http://www.sciencedirect.com/science/article/pii/S0045782503003487.

[4] Bhadeshia HKDH. "Neural networks in materials science". In: *ISIJ international* 39.10 (1999), pp. 966–979.

[5] J Ghaboussi, JH Garrett Jr, and Xiping Wu. "Knowledge-based modeling of material behavior with neural networks". In: *Journal of engineering mechanics* 117.1 (1991), pp. 132–153.

[6] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[7] David McDowell. "A perspective on trends in multiscale plasticity". In: *International Journal of Plasticity* 26 (Sept. 2010), pp. 1280–1309. DOI: 10.1016/j.ijplas.2010.02.008.

[8] I.B.C.M. Rocha, P. Kerfriden, and F.P. van der Meer. "Micromechanics-based surrogate models for the response of composites: A critical comparison between a classical mesoscale constitutive model, hyper-reduction and neural networks". In: *European Journal of Mechanics - A/Solids* 82 (2020), p. 103995. ISSN: 0997-7538. DOI: https://doi.org/10.1016/j.euromechsol.2020.103995. URL: http://www.sciencedirect.com/science/article/pii/S0997753820300565.

[9] Pierre-Alain Guidault et al. "A two-scale approach with homogenization for the computation of cracked structures". In: *Computers & structures* 85.17-18 (2007), pp. 1360–1371.

[10] Marc GD Geers, Varvara G Kouznetsova, and WAM Brekelmans. "Multi-scale computational homogenization: Trends and challenges". In: *Journal of computational and applied mathematics* 234.7 (2010), pp. 2175–2182.

[11] V Kouznetsova, WAM Brekelmans, and FPT Baaijens. "An approach to micro-macro modeling of heterogeneous materials". In: *Computational mechanics* 27.1 (2001), pp. 37–48.

[12] Paul Anthony Sparks. *Reduced order homogenization models for failure of heterogeneous materials*. Vanderbilt University, 2015.

[13] I.B.C.M. Rocha. "Numerical and Experimental Investigation of Hygrothermal Aging in Laminated Composites." In: (2019). URL: https://doi.org/10.4233/uuid:0eab23c7-9ba4-4d27-91ee-58f9f140dd34.

[14] M.A. Bessa et al. "A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality". In: *Computer Methods in Applied Mechanics and Engineering* 320 (2017), pp. 633–667. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2017.03.037. URL: http://www.sciencedirect.com/science/article/pii/S0045782516314803.

[15] M. Vogler, R. Rolfes, and P.P. Camanho. "Modeling the inelastic deformation and fracture of polymer composites – Part I: Plasticity model". In: *Mechanics of Materials* 59 (2013), pp. 50–64. ISSN: 0167-6636. DOI: https://doi.org/10.1016/j.mechmat.2012.12.002. URL: http://www.sciencedirect.com/science/article/pii/S016766361200213X.

[16] FP van der Meer and LJ Sluys. "Continuum Models for the analysis of progressive failure in composite laminates". Undefined/Unknown. In: *Journal of Composite Materials* 43.20 (2009), pp. 2131–2156. ISSN: 0021-9983. DOI: 10.1177/002199830934054.

[17] Frans P Van der Meer. "Mesolevel modeling of failure in composite laminates: constitutive, kinematic and algorithmic aspects". In: *Archives of Computational Methods in Engineering* 19.3 (2012), pp. 381–425.

[18] Oliver Dürr. *Neural Network Architectures*. Last accessed 22 August 2020. 2019. URL: https://freecontent.manning.com/neural-network-architectures/.

[19] M. Nielsen. *A visual proof that neural nets can compute any function*. [Online; accessed 1-December-2019]. 2019. URL: http://neuralnetworksanddeeplearning.com/chap4.html.

[20] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. New York: Springer, 2006. ISBN: 0387310738 9780387310732.

[21] Torgyn Shaikhina and Natalia A Khovanova. "Handling limited datasets with neural networks in medical applications: A small-data approach". In: *Artificial intelligence in medicine* 75 (2017), pp. 51–63.

[22] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (June 2014), pp. 1929–1958.

[23] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. 2015. URL: http://neuralnetworksanddeeplearning.com/.

[24] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[25] WildML. *Artificial Intelligence, Deep Learning, and NLP*. Accessed: 03 February 2020. 2015. URL: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/.

[26] Y. Bengio, P. Simard, and P. Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166.

[27] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. *On the difficulty of training Recurrent Neural Networks*. cite arxiv:1211.5063Comment: Improved description of the exploding gradient problem and description and analysis of the vanishing gradient problem. 2012. URL: http://arxiv.org/abs/1211.5063.

[28] Junyoung Chung et al. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. cite arxiv:1412.3555Comment: Presented in NIPS 2014 Deep Learning and Representation Learning Workshop. 2014. URL: http://arxiv.org/abs/1412.3555.

[29] F. Ghavamian and A. Simone. "Accelerating multiscale finite element simulations of history-dependent materials using a recurrent neural network". In: *Computer Methods in Applied Mechanics and Engineering* 357 (2019), p. 112594. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2019.112594. URL: http://www.sciencedirect.com/science/article/pii/S0045782519304700.

[30] Atsuya Oishi and Genki Yagawa. "Computational mechanics enhanced by deep learning". In: *Computer Methods in Applied Mechanics and Engineering* 327 (2017), pp. 327–351.

[31] M. Mozaffar et al. "Deep learning predicts path-dependent plasticity". In: *Proceedings of the National Academy of Sciences* 116.52 (2019), pp. 26414–26420. ISSN: 0027-8424. DOI: 10.1073/pnas.1911815116. eprint: https://www.pnas.org/content/116/52/26414.full.pdf. URL: https://www.pnas.org/content/116/52/26414.

[32] Z. Liu. *Progresses in deep material networks and materials design*. 15th U.S. National Congress on Computational Mechanics Austin, TX, USA. 2019.

[33] Zeliang Liu and C. T. Wu. "Exploring the 3D architectures of deep material network in data-driven multiscale mechanics." In: *CoRR* abs/1901.04832 (2019). URL: http://dblp.uni-trier.de/db/journals/corr/corr1901.html#abs-1901-04832.

[34] Zeliang Liu et al. "Intelligent multiscale simulation based on process-guided composite database". In: *arXiv preprint arXiv:2003.09491* (2020).

[35] Aditya Khosla et al. “Novel Dataset for Fine-Grained Image Categorization”. In: *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*. Colorado Springs, CO, June 2011.

[36] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. 2014. URL: http://arxiv.org/abs/1412.6980.

[37] *Jive • Dynaflow Research Group*. July 2020. URL: https://software.dynaflow.com/jive/.

[38] A.R. Melro et al. “Micromechanical analysis of polymer composites reinforced by unidirectional fibres: Part I – Constitutive modelling”. In: *International Journal of Solids and Structures* 50.11 (2013), pp. 1897–1905. ISSN: 0020-7683. DOI: https://doi.org/10.1016/j.ijsolstr.2013.02.009. URL: http://www.sciencedirect.com/science/article/pii/S0020768313000747.

[39] A.R. Melro et al. “Micromechanical analysis of polymer composites reinforced by unidirectional fibres: Part II – Micromechanical analyses”. In: *International Journal of Solids and Structures* 50.11 (2013), pp. 1906–1915. ISSN: 0020-7683. DOI: https://doi.org/10.1016/j.ijsolstr.2013.02.007. URL: http://www.sciencedirect.com/science/article/pii/S0020768313000723.

[40] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.

[41] Frans P. van der Meer. “Micromechanical validation of a mesomodel for plasticity in composites”. In: *European Journal of Mechanics - A/Solids* 60 (2016), pp. 58–69. ISSN: 0997-7538. DOI: https://doi.org/10.1016/j.euromechsol.2016.06.008. URL: http://www.sciencedirect.com/science/article/pii/S0997753816300778.

[42] Anna Choromanska et al. “The loss surfaces of multilayer networks”. In: *Artificial intelligence and statistics*. 2015, pp. 192–204.